THE POST CORRESPONDENCE PROBLEM AND EQUALISERS FOR CERTAIN FREE GROUP AND MONOID MORPHISMS

LAURA CIOBANU AND ALAN D. LOGAN

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.

1. Introduction

In this paper we prove results about the classical Post Correspondence Problem (PCP<sub>FM</sub>), which we state in terms of equalisers of free monoid morphisms, and the analogue problem PCP<sub>FG</sub> for free groups ([CMV08], [MNU14]), and we describe the solutions to PCP<sub>FM</sub> and PCP<sub>FG</sub> for certain classes of morphisms. While the classical PCP<sub>FM</sub> is famously undecidable for arbitrary maps of free monoids [Pos46] (see also the survey [HK97] and the recent result of Neary [Nea15]), PCP<sub>FG</sub> for free groups is an important open question [DKLM19, 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 PCP<sub>FM</sub> or PCP<sub>FG</sub>. 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 PCP<sub>FM</sub> for the free monoid of rank two [EKR82], and therefore understanding the solutions to the PCP<sub>FG</sub> for immersions is a significant step towards resolving the PCP<sub>FG</sub> 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 (Appendix 1), so our results concern a significant proportion of instances.

An instance of the PCP<sub>FM</sub> is a tuple \( I = (\Sigma, \Delta, g, h) \), where \( \Sigma, \Delta \) are finite alphabets, \( \Sigma^*, \Delta^* \) are the respective free monoids, and \( g, h : \Sigma^* \to \Delta^* \) are morphisms. The equaliser of \( g, h \) is \( Eq(g, h) := \{ x \in \Sigma^* | g(x) = h(x) \} \). The PCP<sub>FM</sub> is the decision problem:

\[
\text{Given } I = (\Sigma, \Delta, g, h), \text{ is the equaliser } Eq(g, h) \text{ trivial?}
\]

Analogously, an instance of the PCP<sub>FG</sub> 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 PCP<sub>FG</sub> is the decision problem pertaining to the similarly defined \( Eq(g, h) \) in free groups. Beyond PCP<sub>FM</sub>, in this paper we also consider the Algorithmic Equaliser Problem which, for an instance \( I = (\Sigma, \Delta, g, h) \) with \( g, h \) either free monoid (AEP<sub>FM</sub>) or free group (AEP<sub>FG</sub>) morphisms, says:

\[
\text{Given } I = (\Sigma, \Delta, g, h), \text{ output either}
\]

(a) a finite basis for \( Eq(g, h) \), or

(b) an automaton whose language is \( Eq(g, h) \).

2010 Mathematics Subject Classification. 20-06, 20E05, 20F10, 20M05, 68R15.

Key words and phrases. Post Correspondence Problem, free group, free monoid, marked map, immersion.
For free groups these two problems are in fact the same (if $\text{Eq}(g, h)$ is finitely generated), while for free monoids (a) implies (b). Part (a) of the $\text{AEP}_{FM}$ is known to be soluble when $|\Sigma| = 2$ and one of $g$ or $h$ is non-periodic and insoluble otherwise [Hol03, HK97, Corollary 6].

As automata accepting intersections of regular languages are computable, if Part (b) of the $\text{AEP}_{FM}$ is soluble for a class of maps $C$ then we have the following equivalent, but seemingly stronger, result: there exists an algorithm with input a finite set $S : \Sigma^* \rightarrow \Delta^*$ of morphisms from $C$ and output a finite automaton whose language is $\text{Eq}(S) := \cap_{g,h \in S} \text{Eq}(g, h)$. Similarly, bases of intersections of subgroups of free groups are computable, and so we have: if Part (a) of the $\text{AEP}_{FG}$ is soluble for a class of maps $C$ then there exists an algorithm with input a finite set $S : F(\Sigma) \rightarrow F(\Delta)$ of morphisms from $C$ and output a basis for $\text{Eq}(S)$.

A set of words $s \subseteq \Delta^*$ is marked if each $u, v \in s$ is non-empty and starts with a different letter of $\Delta$. A free monoid morphism $f : \Sigma^* \rightarrow \Delta^*$ is marked if the set $f(\Sigma)$ is marked. An immersion of free groups is a morphism $f : F(\Sigma) \rightarrow F(\Delta)$ where the set $f(\Sigma \cup \Sigma^{-1})$ is marked (see Section 3 for equivalent formulations). Halava, Hirvensalo and de Wolf [HHdW01] showed that $\text{PCP}_{FM}$ is decidable for marked morphisms, and inspired by their methods we were able to obtain stronger results (Theorem A) for this kind of map, as well take this result to the world of free groups (Theorem C), where we employ ‘finite state automata’-like objects called Stallings graphs. In fact, our results resolve the simultaneous $\text{PCP}_{FM}$ and $\text{PCP}_{FG}$, which take as input an arbitrary set $S$ of maps (not just two maps $f, g$) and ask the same questions about equalisers as in the classical setting.

**Theorem A.** If $S : \Sigma^* \rightarrow \Delta^*$ is a set of marked morphisms then there exists an alphabet $\Sigma_S$ and a marked morphism $\psi_S : \Sigma_S^* \rightarrow \Sigma^*$ such that $\text{Image}(\psi_S) = \text{Eq}(S)$. Moreover, if $S$ is a finite set then there exists an algorithm with input $S$ and output the marked morphism $\psi_S$.

**Corollary B.** The simultaneous $\text{PCP}_{FM}$ is decidable for marked morphisms of free monoids.

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

**Corollary D.** The simultaneous $\text{PCP}_{FG}$ 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 [Sta_7 Problems P1 & 5] (also [DV96, Problem 6] [Ven02, Conjecture 8.3] [BMS02, Problem F31]). Here $\text{rk}(H)$ stands for the rank, or minimum number of generators, of a subgroup $H$:

**Conjecture 1.1** (The Equalizer Conjecture, 1984). If $g, h : F(\Sigma) \rightarrow F(\Delta)$ are injective morphisms then $\text{rk}(\text{Eq}(g, h)) \leq |\Sigma|$.

This conjecture has its roots in “fixed subgroups” $\text{Fix}(\phi)$ of free group endomorphisms $\phi : F(\Sigma) \rightarrow F(\Sigma)$ (if $\Sigma = \Delta$ then $\text{Fix}(\phi) = \text{Eq}(\phi, \text{id})$), where Bestvina and Handel proved that $\text{rk}(\text{Fix}(\phi)) \leq |\Sigma|$ for $\phi$ an automorphism [BH92], and Imrich and Turner extended this bound to all endomorphisms [IT89]. Bergman further extended this bound to all sets of endomorphisms [Ber99]. Like Bergman’s result, our first corollary of Theorem C considers sets of immersions, which are injective, and answers Conjecture 1.1 for immersions.

**Corollary E.** If $S : F(\Sigma) \rightarrow F(\Delta)$ is a set of immersions then $\text{rk}(\text{Eq}(S)) \leq |\Sigma|$.
In free monoids, although equalisers of injections are free \cite[Corollary 4]{HK97} they are not necessarily regular languages (and hence not necessarily finitely generated) \cite[Example 6]{HK97}. In order to understand equalisers Eq(S) of sets of maps we need to understand intersections in free monoids, and although the intersection $A^* \cap B^*$ of two finitely generated free submonoids is free \cite{Til72} and one can find a regular expression that represents a basis of $A^* \cap B^*$ \cite{BH77}, the intersection is not necessarily finitely generated \cite{Kar84}. The following result is therefore surprising because we have finite generation, even in the intersection Eq(S) = \bigcap_{g,h \in S} Eq(g, h).

**Corollary F.** If $S : \Sigma^* \to \Delta^*$ is a set of marked morphisms then Eq(S) is a free monoid with \( \text{rk}(\text{Eq}(S)) \leq |\Sigma| \).

**The Algorithmic Equaliser Problem.** The AEP$_{FG}$ is insoluble in general, as equalisers are not necessarily finitely generated \cite{Ven02} Section 3, and is an open problem of Stallings’ if both maps are injective \cite{Sta87} Problems P3 & 5]. Our next corollary of Theorem \[A\] therefore resolves this open problem for immersions.

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

The AEP$_{FM}$ is insoluble in general, primarily as equalisers are not necessarily regular languages \cite{ER78} Example 4.6]. Even for maps whose equalisers form regular languages, the problem remains insoluble \cite{KS10}. Another corollary of Theorem \[A\] is the following.

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

In fact, this is a special case of Corollary \[A\] which algorithmically obtains a free basis for Eq(S) (and not just Eq(g, h)).

**Outline of the article.** In Section \[2\] we prove Theorem \[A\] and its corollaries. The remainder of the paper focuses on free groups, where the central result is Theorem \[G\], which is Theorem \[C\] for $|S| = 2$. In Section \[3\] we reformulate immersions in terms of Stallings’ graphs. In Section \[4\] we define the “reduction” $I' = (\Sigma', \Delta', g', h')$ of an instance $I = (\Sigma, \Delta, g, h)$ of the AEP$_{FG}$ for immersions. Repeatedly computing reductions is the key process in our algorithm. In Section \[5\] we prove the process of reduction reduces the “prefix complexity” of an instance (so the word “reduction” makes sense). In Section \[6\] we prove Theorem \[6.2\] 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.

## 2. Marked morphisms in free monoids

In this section we prove Theorem \[A\] and its corollaries. We use the following immediate fact, which shows that $|\Sigma| \leq |\Delta|$, and in particular, we may assume $\Sigma \subseteq \Delta$.

**Lemma 2.1.** Marked morphisms of free monoids are injective.

Consider morphisms $g : \Sigma^*_1 \to \Delta$ and $h : \Sigma^*_2 \to \Delta$. The set of non-empty words over an alphabet $\Sigma$ is denoted $\Sigma^+$. For $a \in \Delta$, a pair $(u, v) \in \Sigma^+ \times \Sigma^+$ 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 $a_i := (u_i, v_i), 1 \leq i \leq m$, then let $\Gamma$ be the alphabet consisting of these blocks and define $g' : (\Gamma^*)^* \to \Sigma^*_1$ by $g'(a_i) = u_i$ and $h' : (\Gamma^*)^* \to \Sigma^*_2$ by $h'(a_i) = v_i$. These maps are computable and, by an identical logic to \cite[Section 2]{HHW01}, are seen to be marked. The map $gg' = hh'$, which we call $k$ (so $k : (\Sigma^*)^* \to \Delta^*$), is the composition of marked morphisms and hence is itself marked. We therefore have the following.
Lemma 2.2. If $g : \Sigma_1^* \to \Delta^*$ and $h : \Sigma_2^* \to \Delta^*$ are marked morphisms then the corresponding maps $g' : \Sigma_1^* \to \Sigma_1^*$, $h' : \Sigma_2^* \to \Sigma_2^*$ and $k : \Sigma_1^* \to \Delta^*$ are marked and are computable.

We require $\Sigma_1 \neq \Sigma_2$ in the proof of Lemma 2.6. Meanwhile, we take $\Sigma_1 = \Sigma = \Sigma_2$.

The reduction of an instance $I = (\Sigma, \Delta, g, h)$ of the marked PCP$_{FM}$, as defined in HHdW01, is the instance $I' = (\Sigma', \Delta, g', h')$ where $\Sigma'$ is defined as above, and where $g'$ and $h'$ are as above, but with codomain $\Delta$ (which we may do as $\Sigma \subseteq \Delta$). We additionally assume that $\Sigma' \subseteq \Sigma$; we can do this as $|\Sigma'| \leq |\Sigma|$ by Lemma 2.2.

The following relies on HHdW01 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 PCP$_{FM}$ are strongly equivalent if their equalisers are isomorphic, which we write as $\text{Eq}(I_1) \cong \text{Eq}(I_2)$.

Lemma 2.3. 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 $g'(|\text{Eq}(I')|) = \text{Eq}(I) = h'(|\text{Eq}(I')|)$.

Proof. Firstly, note that $g'(|\text{Eq}(I')|) \leq \text{Eq}(I)$ HHdW01 Lemma 1, paragraph 2]. From HHdW01 Lemma 1, paragraph 1] it follows that $g'(|\text{Eq}(I')|) \geq \text{Eq}(I)$, so $g'(|\text{Eq}(I')|) = \text{Eq}(I)$. As $g'$ is injective, the map $g'|_{\text{Eq}(I')}$ is an isomorphism. Hence, $I$ and $I'$ are strongly equivalent, and, by symmetry for the $h'$ map, $g'(|\text{Eq}(I')|) = \text{Eq}(I) = h'(|\text{Eq}(I')|)$ as required. \hfill $\square$

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

Theorem 2.4. If $I = (\Sigma, \Delta, g, h)$ is an instance of the marked PCP$_{FM}$ then there exists an alphabet $\Sigma_{g,h}$ and a marked morphism $\psi_{g,h} : \Sigma_{g,h}^* \to \Sigma^*$ such that $\text{Image}(\psi_{g,h}) = \text{Eq}(I)$. 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 HHdW01 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 2.3), we obtain in each case a subset $\Sigma_{g,h}$ (possibly empty) which forms a basis for $\text{Eq}(I_j)$: For Case (1), writing $\Sigma_j = \{a\}$, the result holds as if $g(a) = h(a)$ 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 $\Sigma_j$ forms a basis for $\text{Eq}(I_j)$.

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 2.3 $\text{Eq}(I_j) = g_{j+1}g_{j+2} \cdots g_r(\text{Eq}(I_r)) = \text{Eq}(I_r)$; thus $g_r := g_{j+1}g_{j+2} \cdots g_r$ restricts to an automorphism of $\text{Eq}(I_j)$, so $g_r|_{\text{Eq}(I_j)} \in \text{Aut}(\text{Eq}(I_j))$. The automorphism $g_r$ is necessarily length-preserving ($|g_r(w)| = |w|$ for all $w \in \text{Eq}(I_j)$). Consider $x \in \text{Eq}(I_j) = \text{Eq}(I_r)$. Then $g_r$ maps the letters occurring in $x_r$ to letters and so $g_j(= g_r)$ and $h_j(= h_r)$ map the letters occurring in $x$ to letters, and it follows that every letter occurring in $x$ is a solution to $I_r = I_j$. Hence, a subset $\Sigma_{g,h}$ of $\Sigma_j$ forms a basis for $\text{Eq}(I_j)$ as required.
Therefore, in all three cases a subset \( \Sigma_{g,h} \) of \( \Sigma_j \) forms a basis for \( \text{Eq}(I_j) \), and since \( \Sigma_j \) 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 2.2, 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 \( \text{Image}(\psi_{g,h}) = g_1 g_2 \cdots g_j(\text{Eq}(I_j)) = \text{Eq}(I) \), by Lemma 2.3 and the above, the result follows.

Theorem 2.4 combines with the following general result to give the non-algorithmic part of Theorem A. A subsemigroup \( M \) of a free monoid \( \Sigma^* \) is marked is it is the image of a marked morphism.

**Lemma 2.5.** If \( \{M_j\}_{j \in J} \) is a set of marked subsemigroups of \( \Sigma^* \) then the intersection \( \cap_{j \in J} M_j \) 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 \( \phi \) is a marked morphism. If \( x \) and \( y \) have a nontrivial common prefix, then because \( \phi \) 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 \( x, y \in \cap_{j \in J} M_j \), 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 \( \cap_{j \in J} M_j \). Therefore, \( z_a \) is a prefix of every element of \( \cap_{j \in J} M_j \) beginning with an \( a \). It follows that \( \cap_{j \in J} M_j \) is immersed, as required.

We now prove the algorithmic part of Theorem A (this is independent of Lemma 2.5).

**Lemma 2.6.** There exists an algorithm with input a finite set of marked morphisms \( S : \Sigma^* \to \Delta^* \) and output a marked morphism \( \psi_S : \Sigma_S^* \to \Sigma^* \) such that \( \text{Image}(\psi_S) = \text{Eq}(S) \).

**Proof.** We use induction on \( |S| \). By Theorem 2.4, the result holds if \( |S| = 2 \). Suppose the result holds for all sets of \( n \) marked morphisms, \( n \geq 2 \), 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 \( \text{Image}(\psi_{S_g}) = \text{Eq}(S_g) \) and \( \text{Image}(\psi_{g,h}) = \text{Eq}(g,h) \).

By Lemma 2.2 there exists a (computable) marked morphism \( \psi_S : \Sigma_S^* \to \Sigma^* \) such that \( \text{Image}(\psi_S) = \text{Image}(\psi_{S_g}) \cap \text{Image}(\psi_{g,h}) \) (the map \( \psi_S \) corresponds to the map \( k \) in Lemma 2.2 and \( \Sigma_S \) to \( \Sigma' \)). Then, as required: \( \text{Image}(\psi_S) = \text{Image}(\psi_{S_g}) \cap \text{Image}(\psi_{g,h}) = \text{Eq}(S_g) \cap \text{Eq}(g,h) = \text{Eq}(S) \).

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

**Proof of Theorem A** By applying Lemma 2.5 to Theorem 2.4, there exists an alphabet \( \Sigma_S \) and a marked morphism \( \psi_S : \Sigma_S^* \to \Sigma^* \) such that \( \text{Image}(\psi_S) = \text{Eq}(S) \), while by Lemma 2.6 if \( S \) is finite then such an immersion can be algorithmically found.

We now prove Corollary F which says that \( \text{Eq}(S) \) is free of rank \( \leq |\Sigma| \).

**Proof of Corollary F** Consider the marked morphism \( \psi_S : \Sigma_S^* \to \Sigma^* \) given by Theorem A. By Lemma 2.2, \( \psi_S \) is injective so \( \text{Image}(\psi_S) \) is free. As \( \psi_S \) 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 \( |\Sigma_S| \leq |\Sigma| \) as required.

We now prove a strong form of the AEPFM for marked morphisms.
Corollary 2.7. There exists an algorithm with input a finite set $S : \Sigma^* \to \Delta^*$ of marked morphisms and output a basis for $\text{Eq}(S)$.

Proof. To algorithmically obtain a basis for $\text{Eq}(S)$, first use the algorithm of Theorem A to obtain the marked morphism $\overline{\psi}_S : \Sigma_S^* \to \Sigma^*$ such that $\text{Image}(\overline{\psi}_S) = \text{Eq}(S)$. Then, recalling that we store $\psi_S$ 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 AEP$_{FM}$ for marked morphisms, follows from Corollary 2.7 by taking $|S| = 2$, while Corollary B, the simultaneous PCP$_{FM}$, also follows as $\text{Eq}(S)$ is trivial if and only if its basis is empty.

3. IMMERSIONS OF FREE GROUPS

We denote the free group with finite generating set $\Sigma$ 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 $xx^{-1}$, $x \in \Sigma^\pm 1$, 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 embark on our study of immersions of free groups, as defined in the introduction. We give two alternative characterisations now. The explanation of all the terms involved follows the statement of the lemma.

Lemma 3.1. 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 [Kap00]. 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.

The (unfolded) Stallings graph $\Gamma_g$ 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 $\Delta^{-1}$ occur as edges traversed backwards and we denote by $e^{-1}$ the edge $e$ in opposite direction, and by $E^\pm_1 \Gamma_g$ the sets of edges in both directions. A path $q = (e_1, \ldots, e_n), e_i \in E^\pm_1 \Gamma_g$ edges, is reduced if it has no backtracking, that is, $e_i^{-1} \neq e_{i+1}$ for all $1 \leq i < n$. We denote by $\iota(p)$ the initial vertex of a path $p$ and $\tau(p)$ the terminal vertex, and call a reduced path $p$ with $\iota(p) = u = \tau(p)$ a closed reduced circuit.

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

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

Immersions are precisely those maps $g$ such that every element of $L(\Gamma_g, v_g)$ is freely reduced; this corresponds to the automaton $(\Gamma_g, v_g)$ 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 $(\Gamma_g, v_g)$ is deterministic, $(\Gamma_g, v_g)^{-1}$ is not). For such maps, $L(\Gamma_g, v_g)$ is precisely the image of the map $g$ [KM02, Proposition 3.8].
Example 3.2. 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 \( \Gamma_g \), where the double arrow represents \( x \) and the single arrow represents \( y \), 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 (7)) and \( |g(ba)| = 4 < 6 = |g(a)| + |g(b)| \) (violating Characterisation (3)).

![Figure 1](image)

Using Characterisation (2), we see that immersions are injective [KM02, Proposition 3.8]:

**Lemma 3.3.** 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 PCP\(_{FG}\) 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 PCP\(_{FG}\), which is similar to the reduction in the free monoid case.

Let \( \Gamma \) be a directed, labeled graph and \( u \in VT \) a vertex of \( \Gamma \). The core graph of \( \Gamma \) at \( u \), written \( \text{Core}_u(\Gamma) \), is the maximal subgraph of \( \Gamma \) containing \( u \) but no vertices of degree 1, except possibly \( u \) itself. Note that \( L(\text{Core}_u(\Gamma), u) = L(\Gamma, u) \). For \( \Gamma_1, \Gamma_2 \) directed, labeled graphs, the product graph of \( \Gamma_1 \) and \( \Gamma_2 \), denoted \( \Gamma_1 \otimes \Gamma_2 \), is the subgraph of \( \Gamma_1 \times \Gamma_2 \) with vertex set \( VT_1 \times VT_2 \) and edge set \( \{(e_1, e_2) \mid e_i \in E\Gamma_i, label(e_1) = label(e_2)\} \). One may think of the standard construction of an automaton recognising the intersection of two regular languages, each given by a finite state automaton \( \Gamma_i \), 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 \( \text{Core}(g, h) \), is the core graph of \( g \otimes h \) at the vertex \( v_{g,h} := (v_g, v_h) \), so \( \text{Core}(g, h) := \text{Core}_{v_{g,h}}(\Gamma_g \otimes \Gamma_h) \). We shall refer to \( v_{g,h} \) as the central vertex of \( \text{Core}(g, h) \). Note that \( \text{Core}(g, h) \) represents the intersection of the two images [KM02, Lemma 9.3], in the sense that

\[
L(\text{Core}(g, h), v_{g,h}) = \text{Image}(g) \cap \text{Image}(h).
\]

Write \( \delta_g : \text{Core}(g, h) \to \Gamma_g \) and \( \delta_h : \text{Core}(g, h) \to \Gamma_h \) for the restriction of \( \text{Core}(g, h) \) 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 \( \text{Core}(g, h) \) is a bouquet and every element of \( L(\text{Core}(g, h), v_{g,h}) \) is freely reduced [KM02, Lemma 9.2]. We therefore have free group morphisms \( g' : L(\text{Core}(g, h), v_{g,h}) \to L(\Gamma_g, v_g) \) and \( h' : L(\text{Core}(g, h), v_{g,h}) \to L(\Gamma_h, v_h) \) induced by the maps \( \delta_g, \delta_h \), where \( L(\Gamma_g, v_g) = F(\Sigma_1) \) and \( L(\Gamma_h, v_h) = F(\Sigma_2) \). These maps are computable [KM02, Corollary 9.5]. Let \( \Sigma' \) be the alphabet whose elements consist of the petals of \( \text{Core}(g, h) \). Then \( \Sigma' \) generates the free group \( L(\text{Core}(g, h), v_{g,h}) \), so \( F(\Sigma') = L(\text{Core}(g, h), v_{g,h}) \).
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 4.1.** 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 $k : F(\Sigma) \to F(\Delta)$ are immersions and are computable.

We require that $\Sigma_1 \neq \Sigma_2$ in the proof of Lemma 7.2. Meanwhile, we take $\Sigma_1 = \Sigma = \Sigma_2$.

**Reduction.** The reduction of an immersed instance $I = (\Sigma, \Delta, g, h)$ of the PCP$_{FG}$ is the instance $I' := (\Sigma', \Delta, g', h')$ where $\Sigma'$ is defined as above, and where $g'$ and $h'$ are as above, but with codomain $\Delta$ (which we may do as $\Sigma \subseteq \Delta$). We additionally assume that $\Sigma' \subseteq \Sigma$; we can do this as $|\Sigma'| \leq |\Sigma|$ by Lemma 4.1. As $I$ is immersed, it follows from Lemma 4.1 that $I'$ is also immersed. In the next section we show that the name “reduction” makes sense, as it reduces the “prefix complexity” of instances.

**Example 4.2.** Consider the maps $g, h : F(a, b, c) \to F(x, y, z)$ given by $g(x) = aba^2, g(b) = y^{-1}, g(c) = xzx$ and $h(a) = x, h(b) = yx^2y, h(c) = z$.

Then the graph $\text{Core}(g, h)$ is a bouquet with two petals labelled $xy^2y$ and $zzz$, and $\text{Image}(g) \cap \text{Image}(h) = \{xy^2y, zzx\}$. Moreover, $g(ab^{-1}) = h(ab) = xy^2y$ and $g(z) = h(xzz) = zzx$. 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 PCP$_{FG}$ are strongly equivalent if the equalisers are isomorphic, which we write as $\text{Eq}(I_1) \cong \text{Eq}(I_2)$.

**Lemma 4.3.** 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 $g'(\text{Eq}(I')) = \text{Eq}(I) = h'(\text{Eq}(I'))$.

**Proof.** It is sufficient to prove that $g'|_{\text{Eq}(I')} = \text{Eq}(I)$ and $h'|_{\text{Eq}(I')} = \text{Eq}(I)$.

As $g'$ is an immersion it is injective, by Lemma 3.3. Therefore, $g'|_{\text{Eq}(I')}$ is injective. To see that $\text{Image}(g'|_{\text{Eq}(I')}) \subseteq \text{Eq}(I)$, suppose $x' \in \text{Eq}(I')$. Writing $x := g'(x') = h'(x') = h(x)$ and so $x = g'(x') \in \text{Eq}(I)$, as required.

To see that $\text{Image}(g'|_{\text{Eq}(I')}) \supseteq \text{Eq}(I)$, suppose $x \in \text{Eq}(I)$. Then there exists a path $p_x \in \text{Core}(g, h)$, $\iota(p_x) = v_{g, h} = \tau(p_x)$, such that $\gamma_g \delta_g(p_x) = g(x) = h(x) = \gamma_h \delta_h(p_x)$ [KM02 Proposition 9.4], where $\gamma_g : \Gamma_g \to \Gamma_{\Delta}$ is the canonical morphism of directed, labeled graphs from $\Gamma_g$ to the bouquet $\Gamma_{\Delta}$ with $\Delta$ petals. Hence, writing $x'$ for the element of $F(\Sigma')$ corresponding to $p_x \in L(\text{Core}(g, h), v_{g, h})$, we have that $gg'(x') = g(x) = h(x) = hh'(x)$. As $h$ and $g$ are injective, by Lemma 3.3, 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 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 $\leq n$.

Let $I = (\Sigma, \Delta, g, h)$ be an immersive instance of the PCP$_{FG}$. We define, analogously to [HHdW01 Section 4] (see also [EKRS2]), the prefix complexity $\sigma(I)$ as:

$$\sigma(I) = |\cup_{a \in \Sigma^+} \{x \in F(\Delta) \mid x \text{ is a proper prefix of } g(a)\}| + |\cup_{a \in \Sigma^+} \{x \in F(\Delta) \mid x \text{ is a proper prefix of } h(a)\}|.$$
In the maps in Example 1.2 $\sigma(I) = 10 + 6 = 16$, 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 $\Gamma_g$ and $\Gamma_h$, respectively.

**Lemma 5.1.** Let $I = (\Sigma, \Delta, g, h)$ be an instance of the PCP$_{FG}$ with $g$ and $h$ immersions, and let $I'$ be the reduction of $I$. Then $\sigma(I') \leq \sigma(I)$.

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

By construction, each petal of $\Gamma_g$ and $\Gamma_h$ 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 $\Gamma$. Similarly to $a \in P\Gamma_g$ and $a \in P\Gamma_h$, we map write $a \in P \text{Core}(g, h)$ for the petal in Core$(g, h)$ corresponding to $a \in (\Sigma')^{\pm 1}$. From now on, all paths are assumed to be reduced. Define

$$G = \bigcup_{a\in\Sigma^{\pm 1}} \{p \in \Gamma_g \mid p \text{ is a proper initial subpath of petal } a \in \Gamma_g\},$$

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

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 $a \in P \text{Core}(g, h)$. 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 $\iota(r_q) \in V_h \text{Core}(g, h)$; the paths $q$ and $r_q$ can be seen as “facing” one another on $a$. As $V_g \text{Core}(g, h) \cap V_h \text{Core}(g, h) = \{v_{g, h}\}$, and as $q$ is a proper initial subpath of $a$, the projection $\delta_h(r_q)$ is a non-trivial path in $\Gamma_h$. Note also that $\iota(\delta_h(r_q)) = v_{g, h}$, as $\iota(r_q) \in V_h \text{Core}(g, h)$, 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 $\xi_h$ 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 $\tau(r_p), \tau(r_q) \in V_g \text{Core}(g, h)$, 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 $\delta_g$ and $\delta_h$ agree on $e_p$ and $e_q$, and so as Core$(g, h)$ is a subgraph of $\Gamma_g \times \Gamma_h$ we have that $e_p = e_q$. As 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 $\xi_h$ is injective, and so $|G'| \leq |H|$. The same will hold for an analogously defined function $\xi_H$ from $H'$ to $G$, so $|H'| \leq |G|$. Therefore, $\sigma(I') = |G'| + |H'| \leq |G| + |H| = \sigma(I)$. $\square$

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

**Lemma 5.2.** There exist only finitely many distinct instances $I = (\Sigma, \Delta, g, h)$ of the PCP$_{FG}$ that satisfy $\sigma(I) \leq n$.

As the reduction $I'$ of an instance $I$ gives $\sigma(I') \leq \sigma(I)$, 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 ($\text{AEP}_{\text{FG}}$)

The algorithm for solving the AEP$_{\text{FG}}$ 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 5.2, 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 4.3), we obtain in each case a subset $\Sigma_{g,h}$ (possibly empty) which forms a basis for $\text{Eq}(I_j)$: 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 $x = a_1^{e_1} \cdots a_n^{e_n}$ such that $g(x) = h(x)$. Then as $g$ and $h$ are injective, the words $g(a_i)^{e_i} \cdots g(a_n)^{e_n}$ and $h(a_i)^{e_i} \cdots h(a_n)^{e_n}$ are freely reduced and hence are the same word, and so $g(a_i) = h(a_i)$. The result then follows for Case 2. Case 3 has a more involved proof.

**Lemma 6.1.** Let $I = (\Sigma, \Delta, g, h)$ be an immersive instance of the PCP$_{\text{FG}}$ that starts a cycle (i.e. starting the reduction process with $I$ eventually gives $I$ again). If $\text{Eq}(I)$ is non-trivial then a subset of $\Sigma$ forms a basis for $\text{Eq}(I)$.

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

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

where $I_i = (\Sigma_i, \Delta, g_i, h_i)$. By Lemma 4.3 $\text{Eq}(I_0) = g_1 g_2 \cdots g_r (\text{Eq}(I_r)) = \text{Eq}(I_r)$ and so $\overline{g_r} := g_1 g_2 \cdots g_r$ restricts to an automorphism of $\text{Eq}(I_0)$, that is, $\overline{g_r}|_{\text{Eq}(I_0)} \in \text{Aut}(\text{Eq}(I_0))$. For $\overline{h_r}$ defined analogously, $\overline{h_r}|_{\text{Eq}(I_0)} \in \text{Aut}(\text{Eq}(I_0))$. Write $\text{Eq}(I_k)^{(n)}$ for the set of words in $\text{Eq}(I_k)$ of length precisely $n$, and $\text{Eq}(I_k)^{\leq n}$ for the set of words in $\text{Eq}(I_k)$ of length at most $n$. Consider some $x_0 \in \text{Eq}(I_0)$ and write $x_r := \overline{g_r}^{-1}(x_0)$. Then

$x_0 = g_1 g_2 \cdots g_r (x_r) = \overline{g_r}(x_r),$

$x_0 = h_1 h_2 \cdots h_r (x_r) = \overline{h_r}(x_r).$

By Lemma 4.1 both $g_i$ and $h_i$ are immersions for each $i$, and so by Characterisation 3 of Lemma 3.1 we see that $|g_i(w)| \geq |w|$ for all $w \in F(\Sigma_i)$. Hence, $|x_0| \geq |x_r|$. Therefore, for all $m \geq 1$ the map $\overline{g_r}$ induces a map $\overline{g_r}^{(m)} : \text{Eq}(I_r)^{(m)} \rightarrow \text{Eq}(I_r)^{\leq m}$. Clearly $\overline{g_r}^{(1)}$ is a bijection, and so we inductively see that $\overline{g_r}^{(m)}$ has image $\text{Eq}(I_r)^{(m)}$. Therefore, the automorphism $\overline{g_r}$ of $\text{Eq}(I_0)$ is length-preserving ($|\overline{g_r}(w)| = |w|$ for all $w \in \text{Eq}(I)$), and so maps the letters occurring in $x_r$ to letters. Hence, $g_0(= g_r)$ and $h_0(= h_r)$ map the letters occurring in $x_r$ to letters, and it follows that every letter occurring in $x_r$ is a solution to $I_0$. Hence, a subset $\Sigma_{g,h}$ of $\Sigma$ forms a basis for $\text{Eq}(I_r)$. \hfill \Box

We now prove the central theorem of this article, which gives an algorithm to describe $\text{Eq}(I)$ 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 $f : F(\Sigma) \rightarrow F(\Delta)$ as a list $(f(a))_{a \in \Sigma}$.

**Theorem 6.2.** There exists an algorithm with input an immersive instance $I = (\Sigma, \Delta, g, h)$ of the PCP$_{\text{FG}}$ and output an immersion $\psi_{g,h} : F(\Sigma_{g,h}) \rightarrow F(\Sigma)$ such that $\text{Image}(\psi_{g,h}) = \text{Eq}(I)$. 

Proof. Start by making reductions $I := I_0 \rightarrow I_1 \rightarrow \cdots$. By Lemma 7.2 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 $\Sigma_j$ forms a basis for $\text{Eq}(I_j)$. Since $\Sigma_j$ is computable, this basis is as well.

In order to prove the theorem, it is sufficient to prove that there is a computable immersion $\psi_{g,h} : F(\Sigma_{g,h}) \rightarrow F(\Sigma)$. Consider the map $\tilde{g} = g_1g_2\cdots g_j : F(\Sigma_j) \rightarrow F(\Sigma)$ (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 $\text{Image}(\psi_{g,h}) = g_1g_2\cdots g_j(\text{Eq}(I_j)) = \text{Eq}(I)$, by Lemma 4.3 and the above, the result follows.

We now prove Corollary 7.1 which solves the AEP$_{FG}$ for immersions of free groups.

Proof of Corollary 7.1 To algorithmically obtain a basis for $\text{Eq}(I)$, first obtain the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \rightarrow F(\Sigma)$ given by Theorem 6.2. 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 7 and its corollaries. We first give a general result, from which the non-algorithmic part of Theorem 7 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 7.1 is fundamentally identical to the proof of Lemma 2.5 via Characterisation 1 of Lemma 3.1.

Lemma 7.1. If $\{H_j\}_{j \in I}$ is a set of immersed subgroups of $F(\Sigma)$ then the intersection $\cap_{j \in I} H_j$ is immersed.

The following lemma corresponds to the algorithmic part of Theorem 7. Similar to the above, the proof of the lemma is fundamentally identical to the proof of Lemma 2.6.

Lemma 7.2. There exists an algorithm with input a finite set of immersions $S : F(\Sigma) \rightarrow F(\Delta)$ and output an immersion $\psi_S : F(\Sigma_S) \rightarrow F(\Sigma)$ such that $\text{Image}(\psi_S) = \text{Eq}(S)$.

We now prove Theorem 7 which states that the equaliser is the image of a computable immersion.

Proof of Theorem 7. By Lemma 7.1 there exists an alphabet $\Sigma_S$ and an immersion $\psi_S : F(\Sigma_S) \rightarrow F(\Sigma)$ such that $\text{Image}(\psi_S) = \text{Eq}(S)$, while by Lemma 7.2 if $S$ is finite then such an immersion can be algorithmically found.

We now prove Corollary 7.1 which solves the simultaneous PCP$_{FG}$ for immersions.

Proof of Corollary 7.1 First find a basis for $\text{Eq}(I)$: obtain the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \rightarrow F(\Sigma)$ given by Theorem 7. 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 $\text{Eq}(S)$ is trivial if and only if this basis is empty.

Finally, we prove Corollary 7 which says that $\text{Eq}(S)$ is of rank $\leq |\Sigma|$.

Proof of Corollary 7. Consider the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \rightarrow F(\Sigma)$ given by Theorem 7. As $\text{Image}(\psi_{g,h}) = \text{Eq}(S)$ we have that $\text{rk}(\text{Eq}(S)) \leq |\Sigma_{g,h}|$, while as $\psi_{g,h}$ is an immersion we have that $|\Sigma_{g,h}| \leq |\Sigma|$, and the result follows.
8. Algorithm to Compute the Equaliser

Theorems △ and □ produce the equaliser of a set $S$ of morphisms as the image of a computable map $\psi_S$. For $S = \{g, h\}$, the structure of the algorithm that gives $\psi_S$ (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 $\leq \sigma(I)$, as explained in Section 9.

(1) Input $I = (\Sigma, \Delta, g, h)$.
(2) Set $c := 0$, $i := 0$, $I_0 := I$
(3) Set $M := (|\Delta| + 1)^{|\Sigma|\sigma(I) + 1}$ (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 △ and □); 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 $\text{composition}(B, i)$ (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 $\text{composition}(B, i)$ and terminate.

Procedure $\text{composition}(B, i)$ computes the composition of a map, stored as a list $B$, and the maps obtained in the reduction process, indexed from $i$ downwards.

$\text{composition}(B, i)$

(1) Set $B := g_i(B)$, where $g_i$ is loaded from memory.
(2) $i := i - 1$
(3) If $i \geq 0$, goto (4); else, output $B$.

9. Complexity Analysis

The size of an instance $I = (\Sigma, \Delta, S)$, $S$ a set of morphisms, is $|\Sigma| + |\Delta| + \sum_{g \in S} \sum_{a \in \Sigma^+ 1} |g(a)|$.

The algorithm underlying Theorem △ can be run with $O(2^n)$ space, where $n$ is the size of the input instance $I$, which gives a time bound of $O(2^n)$. The space grows exponentially, unlike in [HHdW01], because the algorithm computes instances that must each be stored (as the immersion $\psi_{g,h}$ is their composition; this corresponds to the function $\text{composition}(B, i)$, above). To obtain this space complexity, first suppose $|S| = 2$ (so consider the function $\text{pairs}(g, h)$, above). There are at most $(|\Delta| + 1)^{|\Sigma|\sigma(I) + 1}$ instances $I_j$ with $\sigma(I_j) \leq \sigma(I)$ [HHdW01]. 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 $1 \leq i < k$ (as these intersect to give $\text{Eq}(S)$), 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 [HHdW01, Section 6]. Hence, the algorithm can be run in $O(2^n)$ space.

Similarly, the algorithm underlying Theorem □ 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 $\sigma(I_j) \leq \sigma(I)$. To see this, write $m := \sigma(I)$ and $d := |\Delta|$. If $I_j = (\Sigma_j, \Delta_j, g_j, h_j)$ is such that $\sigma(I_j) \leq m$ then $|g(a)| \leq m + 1$ for all $a \in \Sigma_j^+$, as $g(a)$ has at most $m$ proper prefixes, and similarly $|h(a)| \leq m + 1$. 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 $(2d)^{2|\Sigma_j|(m + 1)} \leq (2d)^{2|\Sigma|(m + 1)}$ instances that satisfy $\sigma(I) \leq m$. This is $O(2^n)$ as required.
References

[Ber99] George M. Bergman, Supports of derivations, free factorizations, and ranks of fixed subgroups in free groups, Trans. Amer. Math. Soc. 351 (1999), no. 4, 1531–1550. MR 1458296

[BH77] Meera Blattner and Tom Head, Automata that recognize intersections of free submonoids, Information and Control 35 (1977), no. 3, 173–176. MR 460510

[BH92] Mladen Bestvina and Michael Handel, Train tracks and automorphisms of free groups, Ann. of Math. (2) 135 (1992), no. 1, 1–51. MR 1147956

[BMS02] Gilbert Baumslag, Alexei G. Myasnikov, and Vladimir Shpilrain, Open problems in combinatorial group theory. Second edition, Combinatorial and geometric group theory (New York, 2001), Contemp. Math., vol. 296, Amer. Math. Soc., Providence, RI, 2002, online version: http://www.grouptheory.info pp. 1–38. MR 1921705

[CMV08] Laura Ciobanu, Armando Martino, and Enric Ventura, The generic Hanna Neumann Conjecture and Post Correspondence Problem, http://www.eups.upc.es/~ventura/ventura/files/31t.pdf.

[CR06] Laura Ciobanu and Sasa Radomirovic, Restricted walks in regular trees, Electronic J. of Combinatorics 13 (2006), no. R93.

[DKLM19] Volker Diekert, Olga Kharlampovich, Markus Lohrey, and Alexei Myasnikov, Algorithmic problems in group theory, Dagstuhl seminar report 19131 (2019), http://drops.dagstuhl.de/opus/volltexte/2019/11293/pdf/dagrep_v009_i003_p083_19131.pdf.

[DV96] Warren Dicks and Enric Ventura, The group fixed by a family of injective endomorphisms of a free group, Contemporary Mathematics, vol. 195, American Mathematical Society, Providence, RI, 1996. MR 1385923

[EKR82] A. Ehrenfeucht, J. Karhumäki, and G. Rozenberg, The (generalized) Post correspondence problem with lists consisting of two words is decidable, Theoret. Comput. Sci. 21 (1982), no. 2, 119–144. MR 677104

[ER78] A. Ehrenfeucht and G. Rozenberg, Elementary homomorphisms and a solution of the DOL sequence equivalence problem, Theoret. Comput. Sci. 7 (1978), no. 2, 169–183. MR 509015

[HHdW01] Vesa Halava, Mika Hirvensalo, and Ronald de Wolf, Marked PCP is decidable, Theoret. Comput. Sci. 255 (2001), no. 1-2, 193–204. MR 1819073

[HK97] Tero Harju and Juhani Karhumäki, Morphisms, Handbook of formal languages, Vol. 1, Springer, Berlin, 1997, pp. 439–510. MR 1469999

[Hol03] Štefan Holub, Binary equality sets are generated by two words, J. Algebra 259 (2003), no. 1, 1–42. MR 1953706

[IT89] W. Imrich and E. C. Turner, Endomorphisms of free groups and their fixed points, Math. Proc. Cambridge Philos. Soc. 105 (1989), no. 3, 421–422. MR 985677

[Kap00] Ilya Kapovich, Mapping tori of endomorphisms of free groups, Comm. Algebra 28 (2000), no. 6, 2895–2917. MR 1757436

[Kar84] Juhani Karhumäki, A note on intersections of free submonoids of a free monoid, Semigroup Forum 29 (1984), no. 1-2, 183–205. MR 742132

[KM02] Ilya Kapovich and Alexei Myasnikov, Stallings foldings and subgroups of free groups, J. Algebra 248 (2002), no. 2, 608–668. MR 1882114

[KS10] Juhani Karhumäki and Aleksi Saarela, Noneffective regularity of equality languages and bounded delay morphisms, Discrete Math. Theor. Comput. Sci. 12 (2010), no. 4, 9–17. MR 2760332

[MNU14] Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov, The Post correspondence problem in groups, J. Group Theory 17 (2014), no. 6, 991–1008. MR 3276222

[Nea15] Turlough Neary, Undecidability in binary tag systems and the post correspondence problem for five pairs of words, 32nd International Symposium on Theoretical Aspects of Computer Science, LIPIcs. Leibniz Int. Proc. Inform., vol. 30, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015, pp. 649–661. MR 3356447

[Pos46] Emil L. Post, A variant of a recursively unsolvable problem, Bull. Amer. Math. Soc. 52 (1946), 264–268. MR 15343

[Sta87] John R. Stallings, Graphical theory of automorphisms of free groups, Combinatorial group theory and topology (Alta, Utah, 1984), Ann. of Math. Stud., vol. 111, Princeton Univ. Press, Princeton, NJ, 1987, pp. 79–105. MR 895610
10. APPENDIX 1: THE DENSITY OF MARKED MORPHISMS AND IMMERSIONS

In this appendix we show that immersions and marked morphisms are not a negligible set of the entire set of, respectively, free group and free monoid morphisms, but a constant proportion of those.

Suppose \(|\Sigma| = k\) and \(|\Delta| = m\). Any morphism, in a free monoid or free group, can be given by the images of the generators \(\{a_1, \ldots, a_k\}\). Then a morphism \(\phi : \Sigma^* \to \Delta^*\), 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 \(\Delta^*\), and \(\sum_{1 \leq i \leq n} m^i \sim cm^n\) for some constant \(c\). Hence, the number of morphisms from \(\Sigma^*\) to \(\Delta^*\) is \(\sim cm^n\). Counting such words is a lot more delicate than in the monoid case, but the asymptotics are similar, due to the following result ([CR06, Proposition 1]).

Proposition 10.1. Let \(A\) and \(B\) be subsets of \(\Delta^{\pm 1}\). 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

\[
f_{A,B}(n) = \frac{(2m - |A|)(2m - |B|)(2m - 1)^{n-1} + x m + (-1)^n(|A||B| - y m)}{2m} = O((2m - 1)^n),
\]

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

Thus for fixed sets \(A\) and \(B\) the number of immersions with images in the ball of radius \(n\) is \(f_{A,B}(n-2) \sim (c_1(2m - 1)^{n-2})^k\) by the same argument as in the monoid case, for some constant \(c_1\). Since there are only finitely many choices for \(A\) and \(B\), and the number of elements in \(F(\Delta)\) of length \(\leq n\) is \(\sim (c_2(2m - 1)^n)^k\), we get again that the number of immersions over the total number of maps \(F(\Sigma) \to F(\Delta)\) is, as \(n \to \infty\), a positive constant depending on \(k\) and \(m\).
11. Appendix 2: additional material to Sections 2 – 7

Proof of Lemma 7.1. 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.

Proof of Lemma 3.1. (1) Proof of Lemma 2.1. Let $g$ algorithmically obtain immersions $\psi_n$ with base vertex $v$, with base vertex $v$, 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.

Proof of Lemma 7.1. 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 $z_1 z_2 \cdots z_n z'_n + 1$ 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 $x, y \in \cap_{j \in J} H_j$, 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 $\cap_{j \in J} H_j$. Therefore, $z_a$ is a prefix of every element of $\cap_{j \in J} H_j$ beginning with an $a$. It follows that $\cap_{j \in J} H_j$ is immersed, as required.

Proof of Lemma 7.2. We proceed by inducting on $|S|$. By Theorem 6.2, the result holds if $|S| = 2$. Suppose the result holds for all sets of $n$ immersions, $n \geq 2$, 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 $\psi_{S_g} : F(S_g) \to F(\Sigma)$ and $\psi_{g,h} : F(S_{g,h}) \to F(\Sigma)$ such that $\text{Image}(\psi_{S_g}) = \text{Eq}(S_g)$ and $\text{Image}(\psi_{g,h}) = \text{Eq}(g, h)$.

By Lemma 4.1, there exists a (computable) immersion $\psi_S : F(S) \to F(\Sigma)$ such that $\text{Image}(\psi_S) = \text{Image}(\psi_{S_g}) \cap \text{Image}(\psi_{g,h})$ (the map $\psi_S$ corresponds to the map $k$ in the lemma, and $\Sigma_S$ to $\Sigma'$). Then we have the required equality:

$$\text{Image}(\psi_S) = \text{Image}(\psi_{S_g}) \cap \text{Image}(\psi_{g,h}) = \text{Eq}(S_g) \cap \text{Eq}(g, h) = \text{Eq}(S).$$

\[HERIOT-WATT UNIVERSITY, EDINBURGH EH14 4AS, SCOTLAND\]

E-mail address: l.ciobanu@hw.ac.uk

\[HERIOT-WATT UNIVERSITY, EDINBURGH EH14 4AS, SCOTLAND\]

E-mail address: a.logan@hw.ac.uk