Descriptions of Groups using Formal Language Theory

Gabriela Aslı Rino Nesin

A thesis presented for the degree of
Doctor of Philosophy

Department of Computer Science
University of Leicester
August 2015

Abstract

Descriptions of Groups using Formal Language Theory
Gabriela Aslı Rino Nesin

This work treats word problems of finitely generated groups and variations thereof, such as word problems of pairs of groups and irreducible word problems of groups. These problems can be seen as formal languages on the generators of the group and as such they can be members of certain well-known language classes, such as the class of regular, one-counter, context-free, recursively enumerable or recursive languages, or less well known ones such as the class of terminal Petri net languages. We investigate what effect the class of these various problems has on the algebraic structure of the relevant group or groups.

We first generalize some results on pairs of groups, which were previously proven for context-free pairs of groups only. We then proceed to look at irreducible word problems, where our main contribution is the fact that a group for which all irreducible word problems are recursively enumerable must necessarily have solvable word problem. We then investigate groups for which membership of the irreducible word problem in the class of recursively enumerable languages is not independent of generating set. Lastly, we prove that groups whose word problem is a terminal Petri net language are exactly the virtually abelian groups.

Acknowledgements

First and foremost I would like to thank my supervisor, Rick Thomas, for being the most incredibly supportive supervisor I have ever had the chance to witness. He clearly cared as much about my well-being as about my work, and this in turn gave me the trust without which I would not have completed this thesis. I am proud to be his academic descendant.

I would also like to thank my examiners, Alexander for generously offering to examine someone not quite in the same area, and thus providing interesting new ways to look at things, and Mark for the meticulous reading of my thesis, for the interesting viva questions, but also for being an academic inspiration.

The computer science department in Leicester has felt like home almost from day one - that pleasure I owe to all the warm and wonderful people that smiled at me in the hallways, shared tasks such as teaching, open days, and Athena Swan meetings, and helped me through the tough parts. In particular, I must thank Sam, Julien, Octavian, Daniela, Alex (times two!), Kyriakos, Steph, Samaneh, Laura and İlke for all the fun and beers and coffee and whining and ideas-bouncing. There are many more names to be included here but it is not well seen to have one's acknowledgements run longer than the thesis itself.

Lastly, I must go back to origins and thank my family: my mother Manuela, for putting up with the panic-phone-calls when I didn't think I could meet a deadline, my brother Bruno for being generally awesome and understanding what I went through, and my father Ali for confirming I was still on the "right path"!

Contents

Chapter 1

Introduction

One area where computer science interfaces with important concepts in mathematics is in the consideration of word problems of finitely generated groups as formal languages (we are talking here of formal languages over a finite alphabet, so in this thesis all groups will be assumed to be finitely generated unless explicitly stated otherwise). The first questions considered here concerned solvability/decidability; the word problem was shown to be undecidable for finitely presented groups by Novikov [44] and Boone [3]; so a finitely presented group can have a word problem that is recursively enumerable but not recursive. Attention then turned to considering groups with word problems in simpler classes of languages. A characterization of groups with a regular word problem was given in [1] and then the same was done (assuming a subsequent deep result of Dunwoody [9]) for groups with a context-free word problem in [43] by Muller and Schupp.

The computer science side of these questions used algorithmic considerations, definitions of formal languages in terms of automata with varying degrees of memory, but also logic (monadic second-order logic in particular) and string-rewriting.

On the mathematical side of things, apart from the expected algebraic and group theoretic considerations, techniques from geometric and combinatorial

group theory were used, as well as from homology and cohomology theory. The Cayley graph of a group gives a visual representation of the relationships between its elements - as such, many nice properties of the group can be deduced from the shape of this graph. For example, Muller and Schupp's characterisation above essentially uses triangulation of the Cayley graph of a context-free group, and considers its number of ends.

In this thesis we try to continue this tradition, where some of our contributions have a fairly geometric flavour: we investigate the number of ends a pair of groups can have, and we prove a theorem about groups with recursively enumerable irreducible word problems using a geometric property of a generating set, which we call 'loopiness'. We go further and make some comments on separability of Cayley graphs of groups where the irreducible word problem is not well-behaved when it comes to recursive enumerability.

Let us come back to the main question for a moment: given a class of formal languages, what can we say about the groups having a word problem in this class? As a group has many word problems, depending on the generating set we choose, one can expect this study to be quite chaotic. However, there is a way around this: for a class of languages closed under inverse monoid homomorphism, if one of the word problems is a member of the class then all of them are. The classes of languages studied previously - the regular, one-counter, context-free, recursively enumerable and recursive languages - all have this property.

As such, algebraic characterizations have been given for groups with a word problem in these classes. As stated above, in [1] Anisimov showed that a group has regular word problem if and only the group is finite. Higman characterized groups with a recursively enumerable word problem in [23] as those embeddable into finitely presented groups (a result commonly called the Higman embedding theorem), and he and Boone characterized groups with recursive word problem in [4] as groups which can be embedded into a simple group which can itself be embedded into a finitely presented group.

Muller and Schupp's characterisation in [43] of groups with context-free word problem as virtually free groups came next. Then Herbst proved in [21] that groups with a one-counter word problem are exactly the virtually cyclic groups.

In this thesis we look at variations on the word problem and investigate what algebraic properties can be deduced from them.

In Chapter 2 we give the group theoretical background we will need in the thesis. We define Cayley graphs of a group and give a short taste of the theory of ends.

In Chapter 3 we then move on to the formal language side of things, defining the classes of formal languages mentioned above. We summarize the various correspondences which we build on. The closure properties under various language operations of the various classes of languages imply closure of the class of groups with word problem in those classes under various group operations.

The work in Chapter 4 is largely inspired by the work in [5] where the authors look at word problems of pairs of groups. Many of the theorems in that paper are valid for other classes of languages - we restate and sometimes reprove those theorems in the most general setting possible. In the same chapter we present partial results about one-counter pairs of groups, relating the questions with geometric considerations such as Schreier graphs and their number of ends.

In Chapter 5 we consider another variation on word problems: the irreducible word problem. This is the set of elements of a word problem with no subword in the word problem. Unfortunately, we do not know whether the irreducible word problem is as well-behaved as the word problem when it comes to recursive enumerability - there could potentially be a group with an irreducible word problem which is recursively enumerable and another irreducible word problem which is not. Our main result is the fact that if all the irreducible word problems of a group are recursively enumerable, then

the group must have solvable word problem. We also give a partial characterisation of groups whose irreducible word problems are badly behaved, if indeed these groups exist.

In Chapter 6, we go back to the word problem, but change the formal language class we are speaking of. As such, we classify groups whose word problem is a terminal Petri net language as exactly the virtually abelian groups. In the process, we state some relationships with other classes of languages, and give a normal form for Petri nets recognizing the word problem of a virtually abelian group.

We finish by suggesting some possibilities for future research in Chapter 7.

Chapter 2

The basics of group theory and word problems

In this chapter we define monoids and finitely generated groups, describing the relationship between them. We focus shortly on some special kinds of groups that we will consistently encounter throughout this thesis, as well as some algebraic constructions on groups - essentially ways to get new groups from old ones. These constructions will be associated in Chapter 3 with invariance properties of certain classes of formal languages, giving a practical way to go back and forth between the two areas.

Finally, we introduce Cayley graphs as a geometric description of a group with respect to a finite generating set. This turns out to be an easy way to visualize a word problem.

2.1 Basics of group theory

2.1.1 Monoids, groups and group presentations

Definition 1. *A* monoid *is a set M closed under a binary operation $\cdot$ which is associative and has an identity element 1_M.*

Given a finite set (or alphabet) Σ, the set of all finite strings of symbols of Σ (called *words*) forms a monoid under the concatenation operation (appending one string onto the end of another). Its identity element is the empty word, denoted λ. This monoid is called the *free monoid* over Σ and is denoted by Σ^*. A (formal) language is just a subset of Σ^* for some finite alphabet Σ.

Definition 2. *A monoid G is called a* group *if every element of G has an inverse, i.e. if for all $g \in G$ there is an $h \in G$ such that $g \cdot h = h \cdot g = 1_G$.*

Often the operation symbol $\cdot$ is omitted (emphasizing the relationship with strings), and the element h as in the definition above is denoted g^{-1}.

Definition 3. *A subset H of G is called a* subgroup *of G if it is a group and the operation on it is the restriction of the operation on G. If H is furthermore not equal to G then it is called a* proper subgroup *of G.*

It follows automatically that in this case, the identity and inverses of elements of H are the same as those of G. We write $H \leqslant G$.

Definition 4. *A group G is said to be* finitely generated *if there is a finite subset A of G such that no proper subgroup of G contains A.*

This means that any element of G can be expressed as a product of elements of A and their inverses. If instead of seeing generators a and a^{-1} as elements of the group we take them as formal symbols, then words over the alphabet $\Sigma = A \cup A^{-1}$ represent elements of G. This the idea behind the following formal definition.

Definition 5. *If a group G is generated by a finite set A then there is an onto monoid homomorphism $\phi : (A \cup A^{-1})^* \to G$. The set A^{-1} is a set of formal symbols $\{a^{-1} : a \in A\}$ with the property that $\phi(a^{-1}) = \phi(a)^{-1}$ for all $a \in A$. We call A a* group generating set *of the group G, and ϕ the* presentation map.

By "a represents g" we shall mean $\phi(a) = g$. Since the definition of ϕ is somehow "natural" given A, we will often use the two interchangeably, assuming ϕ given A or vice versa. In this thesis all groups will be assumed to be finitely generated unless stated otherwise. For all words $w, v \in (A \cup A^{-1})^*$ and any group element $g \in G$, we will write $w =_G g$ to mean $\phi(w) = g$ and write $w =_G v$ to mean $\phi(w) = \phi(v)$; in the second case we say that $w = v$ is a *relation* in G. If we have a relation of the form $w = 1_G$ (i.e. $w =_G \lambda$) we say that w is a *relator* in G.

There are relations, such as $aa^{-1} = 1_G$ for a generator a, which hold in any group. These are the relations which follow directly from the definition of a group. A group where the only relations that hold are the ones that follow from the group definition is called a *free group*. We will see more about these groups later. However, not all groups are free: other relations might hold depending on the group. For example, in the group of rotations of the (labelled) square, generated by the 90 degree rotation r, the relation $r^4 =_G 1_G$ holds, as doing the rotation r four times gives us back the original square. This leads us to the concept of group presentations:

Definition 6. *We say that $\langle A|R\rangle$ is a* presentation *for a group G (and write $G = \langle A|R\rangle$) if A is a generating set for G, R is a set of elements of $\Sigma^* = (A \cup A^{-1})^*$, and any relation holding in G can be deduced from the group definition and by rewriting elements in R to 1_G or vice versa. R is called the set of* relators *of the presentation.*

The above definition is a bit vague: we will give a better one once we have defined normal subgroups and quotients of groups. In any case, $\langle A|R\rangle$ is the most general group generated by A such that the equalities $r =_G 1_G$ for all $r \in R$ hold (together with all of their consequences). Note that the terminology is consistent with that of 'presentation maps': relators are just words which ϕ sends to the identity. The free group over the set A is just the group with presentation $\langle A| \quad \rangle$. The group of rotations of the square can be presented as $\langle r|r^4\rangle$.

In general in group theory, groups are said to be "the same" if they are the same up to isomorphism - homomorphisms which are also bijective.

Definition 7. *Let G and H be two groups, with $\cdot_G$ and $\cdot_H$ as respective operations and 1_G and 1_H as respective identity elements. A map $f : G \to H$ is said to be an* isomorphism *if it is one-to-one, onto, and satisfies*

$$f(g \cdot_G g') = f(g) \cdot_H f(g')$$

for all $g, g' \in G$. We say that G is isomorphic *to H, denoted $G \simeq H$.*

It is straightforward to show that $f(1_G) = 1_H$ and $f(g)^{-1} = f(g^{-1})$. It is also not difficult to show that being isomorphic is an *equivalence relation* - reflexive, symmetric and transitive on the class of groups.

As examples of isomorphic groups, we can see that for generating sets A and B of the same cardinality, the groups $\langle A| \quad \rangle$ and $\langle B| \quad \rangle$ are isomorphic - essentially the same group up to renaming of generators. Similarly, a group may have more than one presentation (i.e. different presentations may give isomorphic groups). For example, consider the groups $\langle a| \quad \rangle$ and $\langle b, c|b^2c^{-3}, bcb^{-1}c^{-1}\rangle$. These two groups are isomorphic via the isomorphism $f(c) = a^2$ and $f(b) = a^3$.

We already know that a group $G = \langle A|R\rangle$ is finitely generated if A is finite. We say that it is *finitely presented* if moreover R is finite.

2.1.2 Normal subgroups, quotients and index

Given a group and a subgroup, we define a notion of how "large" the subgroup is relative to the group. To this end, we define the notion of *coset*. For a subgroup $K \leqslant G$ and an element $g \in G$, define

$$Kg = \{kg : k \in K\}.$$

This is called a *right coset* of K in G (*left cosets* are defined analogously). It is not difficult to prove that for any $g, g' \in G$, Kg and Kg' are either disjoint or equal. Hence, the cosets form a partition of G. The number of right cosets of K making up G is called the *index of K in G* (if G is a union of n right cosets, it is also a union of n left cosets). The index is denoted by $[G : K]$ and can be either finite or infinite. If it is finite, then K is a *finite index subgroup of G*, and G is a *finite extension* of K.

We have the following theorem, whose proof is straightforward:

Theorem 8. *If $K \leqslant H \leqslant G$, then*

$$[G : K] = [G : H][H : K].$$

A subgroup where any left coset gK is equal to the right coset Kg is called a *normal subgroup*, denoted $N \trianglelefteq G$:

Definition 9. *A subgroup N of G is called* normal *if for all $g \in G$, $Ng = gN$.*

The kernel of a homomorphism is always a normal subgroup, and conversely every normal subgroup is the kernel of a homomorphism. For a group homomorphism $f : G \to H$, the *kernel of f* is the set of all elements the function f sends to the identity:

$$Ker(f) := \{g \in G : f(g) = 1_H\}.$$

It is straightforward to show that $Ker(f)$ is a subgroup of G. Now to show it is normal: for any $g \in G$, $g' \in Ker(f)$,

$$f(g^{-1}g'g) = f(g)^{-1}f(g')f(g) = f(g)^{-1}1_H f(g) = 1_H,$$

so $g^{-1}Ker(f)g \subseteq Ker(f)$. Replacing g by g^{-1} and vice versa, we get the opposite inclusion.

Now, when we have a group G with a normal subgroup N, the cosets of N in G form a group themselves.

Definition 10. *If $N \trianglelefteq G$, then G/N is the group whose elements are the cosets Ng (sometimes denoted $[g]$) and with group multiplication*

$$[g][g'] = [gg'].$$

It is easy to see that the identity element is $[1_G]$. This group is called the quotient *of G by N.*

Thus, if N has finite index in G, then G/N is a finite group. Essentially, the quotient is the group where we have identified all elements which are in the same coset. Similarly, if we have a group homomorphism $f : G \to H$, then $G/Ker(f)$ is a 'copy' of G where we have identified the elements which have the same image under f. Furthermore, if f is an onto group homomorphism, then $G/Ker(f) \simeq H$.

By analogy with the above, we can also talk about the quotient of a group (or indeed a monoid) by a congruence $\sim$ (a *congruence* is just an equivalence relation which agrees with the group operation, so that if $m_1 \sim n_1$ and $m_2 \sim n_2$ then $m_1 \cdot m_2 \sim n_1 \cdot n_2$). In that case, for any element m of the monoid M, $[m]_\sim = \{n \in M : m \sim n\}$ (hence the notational similarity to the definition above).

We can now define group presentations properly:

Definition 11. *We say $\langle A|R\rangle$ is a* presentation *for a group G (and write $G = \langle A|R\rangle$) if A is a generating set for G, R is a set of elements of $\Sigma^* = (A \cup A^{-1})^*$, and G is isomorphic to the quotient $F/Ker(\pi)$ where:*

- *F is the free group $\langle A| \quad \rangle$generated by A, and*
- *$\pi : F \to G$ is the group homomorphism sending each element of R to the identity.*

2.1.3 Special kinds of groups and operations on groups

There are many ways to construct new groups from old ones, easily expressible in terms of presentations.

We have seen the definition of a free group, and noted that for every cardinality of generating set there is one free group up to isomorphism. Other types of groups which will be useful in this thesis are the following:

Definition 12. *A* cyclic group *is a group generated by one element only.*

Note that in such a group everything is determined by the *order* of the generator, i.e. how many times it must be multiplied by itself to give the identity (if such a number exists). Hence for example $\langle a|a^3\rangle$ is a cyclic group of order three, with only three elements: the identity, a, and a^2. If the generator does not have finite order, we get $\langle a| \ \rangle$, which is also the free group on one generator, and is often denoted by $\mathbb{Z}$. It is fairly easy to prove that given a value in $n \in \mathbb{N} \cup \{\infty\}$, the cyclic group of order n (i.e. the group generated by a single element of order n) is unique up to isomorphism. The cyclic group of finite order n will often be denoted by either $\mathbb{Z}_n$ or $\mathbb{Z}/n\mathbb{Z}$.

If an element of a group has finite order then it is called a *torsion element*. A *torsion-free* group is one which has no torsion elements. As a side note: surprisingly, a finitely generated group where all elements are torsion can still be infinite - this was an famous problem called the general Burnside problem which stayed open for six decades until Golod and Shafarevitch answered it in the negative in 1964 in [16] (see also Tarski monster groups [45] which constitute an interesting class of examples of these types of groups).

Another type of group where elements interact in a specific way is the following:

Definition 13. *An* abelian group *is a group G where for all generators a and b, $ab =_G ba$.*

Note that this means that in an abelian group any two elements commute, as every element has an expression as products of generators and their

inverses.

We will be particularly interested here in groups which have large free, cyclic or abelian subgroups. In general, we have:

Definition 14. *If P is a property of groups and G is a group with a subgroup H of finite index such that H has property P, then G is said to be* virtually P.

Thus groups with finite index free, cyclic or abelian subgroups are called virtually free, virtually cyclic and virtually abelian respectively.

We have seen above how to get a new group from an old one by quotienting out a normal subgroup. This, in analogy to division, gives us a "smaller" group. We will now see ways of combining two groups into a "larger" one.

Definition 15. *Let $G_1 = \langle A_1 | R_1 \rangle$ and $G_2 = \langle A_2 | R_2 \rangle$ be two groups such that $A_1 \cap A_2 = \emptyset$. The* free product *of G_1 and G_2, denoted $G_1 * G_2$, is the group with presentation $\langle A_1 \sqcup A_2 | R_1 \cup R_2 \rangle$. This can also be generalised to arbitrarily many groups $\{G_i : i \in I\}$, in which case we write the free product as $*_{i \in I} G_i$.*

Thus in the free product, the old generators interact as they used to, but no interaction between the generators of the different component groups occurs. On the other hand, in the direct product, we stipulate that the generators of the factors commute with each other (but there are no other interactions). Thus:

Definition 16. *Let $G_1 = \langle A_1 | R_1 \rangle$ and $G_2 = \langle A_2 | R_2 \rangle$ be two groups such that $A_1 \cap A_2 = \emptyset$. The* direct product *of G_1 and G_2, denoted $G_1 \times G_2$, is the group with presentation $\langle A_1 \sqcup A_2 | R_1 \cup R_2 \cup [A_1, A_2] \rangle$, where*

$$[A_1, A_2] = \{a_1^{-1} a_2^{-1} a_1 a_2 : a_1 \in A_1, a_2 \in A_2\}.$$

The free product is in some sense pasting two groups together without letting them interact (or rather, pasting a copy of one group at each element

of the other, and vice-versa). If we have two groups that have very similar - isomorphic - subgroups, we can also paste them together, but identifying the two isomorphic subgroups.

Definition 17. *Let G and H be two groups with subgroups K_1 and K_2 respectively, such that $\psi : K_1 \to K_2$ is an isomorphism. Let N be the smallest normal subgroup of $G * H$ containing $k\psi(k)^{-1}$ for each $k \in K_1$. Then the* free product with amalgamation $G *_\psi H$ *with respect to ψ is $(G * H)/N$.*

In general, ψ is understood, K_1 and K_2 are identified as K, and the free product with amalgamation is said to be the free product with K amalgamated *and denoted $G *_K H$.*

A related concept is the HNN extension, where instead of two groups we have a single group with two isomorphic subgroups. We paste a new copy of the group to itself, identifying the two copies of the subgroup with the aid of a new generator.

Definition 18. *Let $G = \langle A|R\rangle$ be a group with two subgroups K_1 and K_2, such that $\psi : K_1 \to K_2$ is an isomorphism. Then the* HNN extension of G *relative to ψ, denoted $*_\psi G$, is the group with presentation*

$$\langle A, t|(t^{-1}kt\psi(k)^{-1}) \quad \forall k \in K_1\rangle.$$

Again, when ψ is understood, we write $*_K G$.

Many of the above constructions can be expressed in terms of category theory: the free group on the generating set A is the image of the set A under the free functor for the category of groups. The direct and free products are respectively the product and coproduct in the category of groups. The free product with amalgamation is a pushout in the same category.

Lastly, we shortly mention the *semidirect product*, similar to the direct product but making one of the component groups normal in the resulting group. Formally, $G = N \rtimes H$ if and only if $N \trianglelefteq G$, $G = NH$ and $N \cap H = \{1\}$. Note that H and G/N are then isomorphic.

2.2 Cayley graphs and word problems

Let us return to generating sets of groups. The presentation map ϕ is onto but never one-to-one: there are many words representing the same group element (at the very least, $\phi(aa^{-1}) = 1 = \phi(bb^{-1})$ for any a, b in the generating set). The question may therefore be asked as to which words represent the same elements of the group. Because $\phi(a)^{-1} = \phi(a^{-1})$ for any generator,

$$\phi(w) = \phi(v) \iff \phi(wv^{-1}) = 1$$

for all words w, v (v^{-1} being just what one might expect - if $v = a_1 \dots a_n$ then $v^{-1} = a_n^{-1} \dots a_1^{-1}$), and the above reduces to the simpler question of which words represent the identity.

Definition 19. *Let A be a finite generating set for G. Then the* word problem of G with respect to A *is the set*

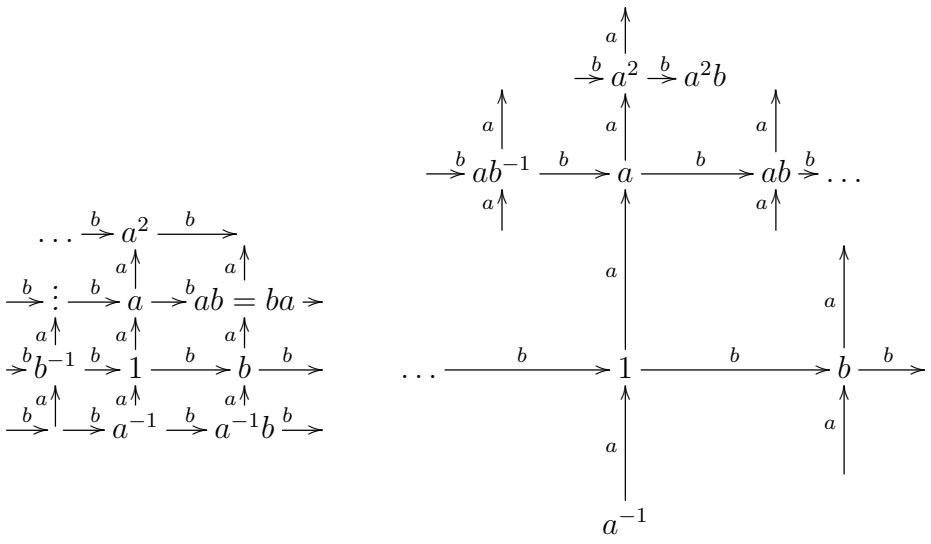
$$W_A(G) = \{w \in (A \cup A^{-1})^* : \phi(w) = 1_G\} = \phi^{-1}(\{1_G\}).$$

Related to the word problem is the *Cayley graph* of a group with respect to a generating set, a way of visualising the structure of a group.

Definition 20. *Let G be a group and A a group generating set for G. The* Cayley graph *of G over A is a labelled oriented graph where the set of nodes is the set of elements of G, and $g \xrightarrow{a} h$ for $a \in A \cup A^{-1}$ if $g\phi(a) = h$. We denote it by $\Gamma(G, A)$.*

Throughout this thesis A will always be finite, so that in our Cayley graphs all vertices will always have finite degree.

Example 21. *Portions of the Cayley graphs of $\mathbb{Z} \times \mathbb{Z} = \langle a, b | aba^{-1}b^{-1} \rangle$ and of the free group of rank two $F_2 := \langle a, b | \quad \rangle$ are depicted below. Only positive labels are displayed - the inverse arrows go the opposite way.*



There is a close relationship between $\Gamma(G, A)$ and $W_A(G)$. Take a word w in $W_G(A)$, and starting at the node in $\Gamma(G, A)$ representing the identity 1_G, start reading each letter of w along the arrows of the graph. Since

$$1_G\phi(w) = 1_G \iff w \in W_A(G),$$

after reading all of w we must be back at the origin node, and conversely the label of a loop starting and ending at 1_G must be a word in the word problem. Hence the word problem with respect to A can be seen as the set of labels of loops in the Cayley graph starting and ending at 1_G.

The reader may have noticed that the shape of the Cayley graph changes according to the generating set one takes for the group (for example, the Cayley graph of $\mathbb{Z}$ under the presentations $\langle a| \quad \rangle$ and $\langle a, b|a^2 = b^3, ab = ba\rangle$) are quite different. However, there is a geometric invariant to the general shape of the Cayley graph: how it looks "from very far away". This invariant is called the *number of ends* of a graph.

Before defining this, we need to define a concept of 'distance' in a Cayley graph. For a directed graph Γ and two nodes x and y, we define the distance $d_\Gamma(x, y)$ between them to be the length of the shortest directed path between them. However, since in a Cayley graph for any edge labelled a from vertex x to vertex y we have another edge labelled a^{-1} from y to x, we do not need to specify that the path be directed. The distance function defined above then truly becomes a metric: symmetric and satisfying the triangular inequality.

Definition 22. *Let Γ be a graph with a designated origin vertex o. Define for every $n \in \mathbb{N}$ the* ball *with centre o and radius n as:*

$$B(o, n) = \{x \in V(\Gamma) : d_\Gamma(o, x) \leqslant n\}.$$

Definition 23. *Let Γ be a connected locally finite graph (i.e. a connected graph where any node has only finitely many edges connected to it) and let o be a vertex designated as the origin. Let $B(o, n)$ be as in Definition 22 above. Define $\Gamma^{(n)} := \Gamma - B(o, n)$ and denote by $cc\left(\Gamma^{(n)}\right)$ the number of connected components of $\Gamma^{(n)}$. Then the* number of ends *of Γ is*

$$e(\Gamma) := \lim_{n\to\infty} cc\left(\Gamma^{(n)}\right).$$

It is not difficult to see that the sequence has a limit: instead of connected components of $\Gamma^{(n)}$, consider infinite connected components only. The limit of that sequence as n goes to infinity is the same as the limit of $\{cc(\Gamma^{(n)})\}_n$, as finite connected components will eventually be fully contained inside a ball and taken away. The sequence of infinite connected components is strictly non-decreasing as n goes to infinity: each successive removal of a bigger ball either disconnects an existing infinite component into two components, at least one of which is infinite, or leaves the infinite component connected. Hence the limit $e(\Gamma)$ exists, though it may be infinite. See [41] for example.

We state the following theorem and only sketch its proof for now - a full proof will be given in Chapter 4.

Theorem 24. *If A and B are two finite generating sets for the same group G, then $\Gamma(G, A)$ and $\Gamma(G, B)$ have the same number of ends.*

Proof Sketch: The "translation" of elements of $A \cup A^{-1}$ into words over the alphabet B have a maximal length. Therefore, when changing generating sets for the Cayley graph, the distance between two points cannot increase or decrease by more than a constant factor (we then call the two graphs 'quasi-isometric'). Therefore the number of ends cannot increase or decrease. □

We will see more about ends of graphs and of Cayley graphs in particular in Chapter 4.

Chapter 3

Formal languages: correspondences with word problems and groups

Here we introduce the five main classes of languages we shall be referring to in this thesis: the classes of regular, one-counter, context-free, recursive and recursively enumerable languages. All of these will be given automata-theoretic definitions. It is also possible to define some of them in terms of generating grammars, however we find the automata-theoretic definitions more useful for the purposes of this thesis.

After describing these classes of formal languages, we make the connection with groups clear. Given a group G and a finite generating set A for it, we see that the word problem of G with respect to A is a formal language over the alphabet $\Sigma = A \cup A^{-1}$. The word problem, as it determines which words represent the same group element, gives us information on the interaction of elements within the group. It should come as no surprise, then, that the study of how complicated a word problem is would give us some information about the algebraic structure of the group: we will see later that there are various correspondences between the algebraic structure of a group and the

formal language class its word problems are in.

We shall make clear why we speak in particular about regular, one-counter, context-free, recursive and recursively enumerable languages. These classes are special because of the various closure properties they enjoy. Next, we will explain why choosing these classes is an advantage when considering groups. In particular, the analysis of group structure as relating to word problems would be quite chaotic if it were the case that a group could have one word problem in a particular class of languages and another word problem not in that class.

3.1 Automata and formal languages

As we mentioned in Chapter 2, a *(formal) language* L over an alphabet Σ is just a subset of the free monoid Σ^*. Central to the study of formal languages is the study of how complicated they are, and this is measured by the model of computation required to 'recognize' such a language. Essentially, given a computational model, for example a type of automaton which is designed to give an answer 'yes' or 'no' on an input word, the question is whether we can construct an automaton of that type which will answer 'yes' on all words of L and 'no' on all other words. We say then that the automaton *recognizes* L. Thus we classify formal languages by the types of automata which are capable of recognising them.

The simplest type of automaton is the (non-deterministic) finite automaton, called an NFA: it has a finite number of states, and transitions labelled by letters of a finite alphabet. A start state determines where we should start reading the input word, and a finite number of final states determine where it should end in order to be accepted. Thus, a word is accepted by the automaton if there is a path through the automaton starting at the start state, ending at a final state, and whose sequence of transitions spells out the input word. The formal definition is given below.

Definition 25. *A* (non-deterministic) finite automaton *or NFA* $\mathfrak{A}$ *is a quintuple* $(Q, \Sigma, \delta, s, F)$ *where*

- Q *is a finite set* (of states),
- Σ *is a finite set* (of input symbols),
- $\delta \subseteq Q \times (\Sigma \cup \{\lambda\}) \times Q$ *is a ternary relation* (the transition relation),
- $s \in Q$ *is a designated state* (the start state) *and*
- $F \subseteq Q$ *is a designated finite set of states* (the final or accept states).

The automaton $\mathfrak{A}$ *is said to* accept *or* recognize *a word* $w \in \Sigma^*$ *if* (s, w, f) *is in the reflexive transitive closure of* δ *for some* $f \in F$*, and is said to* reject *it otherwise. The set of words accepted by* $\mathfrak{A}$ *is denoted* $\mathfrak{L}(\mathfrak{A})$.

Definition 26. *A formal language* $L \subseteq \Sigma^*$ *is said to be* regular *if there is a finite automaton* $\mathfrak{A}$ *such that* $L = \mathfrak{L}(\mathfrak{A})$.

Note that this model is non-deterministic - at a state q, there may be more than one outgoing transition with the same label. However, it turns out that making the model deterministic (i.e. making δ a function rather than a relation and prohibiting transitions labelled by λ) does not change the expressiveness: for every non-deterministic finite automaton there is a deterministic one (called a DFA) recognizing the same language, so we still obtain the same class of languages.

Definition 27. *We denote by* $\mathcal{REG}$ *the class of all regular languages.*

A finite automaton has no memory, in the sense that at any given point it doesn't have any more information than the state it is in. As such, the expressiveness of such a formalism is restricted. For example, a finite automaton cannot "count" beyond its set of states, so to speak. Hence the language $\{a^n b^n : n \in \mathbb{N}\}$ over two symbols a and b is not regular, because no

finite automaton can keep track of the number of as long enough to compare it to the number of bs. To remedy this situation, we can add some memory to the finite automaton. Let us start with the most general way of adding memory: we add to the finite automaton a 'last in first out' stack, which can contain a certain number of stack symbols, and a bottom marker at its very bottom. The automaton then acts more or less like a finite automaton, except that now the possibility of taking a transition at a state can be restricted by the symbol on the top of the stack. Furthermore, at each transition we are allowed to push and pop some symbols on and off the top of the stack (see the rules governing this below).

Definition 28. *A* pushdown automaton $\mathfrak{A}$ *is a sextuple* $(Q, \Sigma, \Gamma, \delta, s, F)$ *where*

- Q *is a finite set of* states*,*
- Σ *is a finite set of* input symbols*,*
- Γ *is a finite set of* stack symbols*, including the* bottom marker $\#$*,*
- $\delta \subseteq Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q \times \Gamma^*$ *is the* transition relation[1]*,*
- $s \in Q$ *is a designated state (the* start state*) and*
- $F \subseteq Q$ *is a designated set of states (the* final *or accept states).*

We require it to satisfy

$$(q, a, \#, r, \gamma) \in \delta \Rightarrow \gamma \in \{\Gamma - \{\#\}\}^*\#$$

and

$$(q, a, g, r, \gamma) \in \delta, g \in (\Gamma \cup \{\lambda\}) - \{\#\} \Rightarrow \gamma \in \{\Gamma - \{\#\}\}^*.$$

[1]Essentially, this is a tuple consisting of the current state, the symbol we're reading, the top of the stack (which we pop off), the next state, and the symbols we will push onto the stack.

In other words, the stack always contains # at the bottom only, followed by a certain number of stack symbols other than #.

A configuration *is a triple in $Q \times \Sigma^* \times \{\Gamma - \{\#\}\}^*\#$, keeping track of the state we are currently in, the portion of the input word remaining to be read and the current configuration of the stack (read from top to bottom).*

We inductively define a binary relation $\vDash$ on the configurations as follows:

$$(q, a\beta, g\gamma) \vDash (r, \beta, \theta\gamma) \text{ if } a \in \Sigma \text{ and } (q, a, g, r, \theta) \in \delta$$

and

$$(q, \beta, g\gamma) \vDash (r, \beta, \theta\gamma) \text{ if } (q, \lambda, g, r, \theta) \in \delta.$$

Writing $\vDash^*$ for the reflexive transitive closure of $\vDash$, we then say that $\mathfrak{A}$ accepts *a word α if*

$$(s, \alpha, \#) \vDash^* (f, \lambda, \gamma\#)$$

for some $\gamma \in \{\Gamma - \{\#\}\}^*$, $f \in F$.

Note that the acceptance condition is the same as that for a finite automaton - there is no need for the stack to be empty when we have finished reading the input word. Again, this does not make a difference in terms of computational power; for every pushdown automaton that accepts by final state only, we can construct a pushdown automaton that accepts by final state and empty stack by adding to every final state a λ-transition (a transition reading nothing) that exclusively pops symbols off the stack until it reaches the bottom marker.

Definition 29. *A formal language L is said to be* context-free *if there is a pushdown automaton $\mathfrak{A}$ such that $L = \mathfrak{L}(\mathfrak{A})$. The class of all context-free languages is denoted by $\mathcal{CF}$.*

The augmentation in memory brings a great increase in expressiveness: for example one can now recognize the language $L = \{a^n b^n : n \in \mathbb{N}\}$. But in fact we don't even need all this memory in order to recognize L; a single

stack symbol would suffice. We would just have to push the symbol onto the stack each time an a is read, and to pop it off each time a b is read. With this in mind, we make the following definition:

Definition 30. *A* one-counter automaton *is a pushdown automaton whose stack alphabet $\Gamma = \{\#, c\}$ contains only one symbol c apart from the bottom marker.*

For all intents and purposes, in a one-counter automaton the stack can be represented simply by a natural number. This model is strictly weaker than a pushdown automaton. Although we can now recognize languages such as $\{a^n b^n : n \in \mathbb{N}\}$, we can still only recognize languages where we are counting one thing at a time (or comparing, such as the number of as and bs above). For example, consider the language $\{a^i b^j c^j d^i : i, j \in \mathbb{N}\}$ over four symbols a, b, c, d: a one-counter automaton is too weak to recognize it. Intuitively, this is because we have to choose whether to keep track of the number of as or bs in the stack initially; if we don't, then the only thing we will be keeping track of is the sum $i + j$, and this level of detail is not enough (we will end up accepting a word not in L). We need more than one counter to recognize this language - we in fact need the full power of a pushdown automaton to recognize it (intuitively, push a certain symbol while reading as, then a different symbol when reading bs. We can now match the cs with the bs and the ds with the as by popping them off the stack due to the use of two different symbols).

Definition 31. *A formal language L is said to be* one-counter *if there is a one-counter automaton $\mathfrak{A}$ such that $L = \mathfrak{L}(\mathfrak{A})$. The class of all one-counter languages is denoted by $\mathcal{OC}$.*

Lastly, we define recursive and recursively enumerable languages. These are often more formally defined in terms of Turing machines accepting them - as such they are more powerful than all the classes of automata previously mentioned. However, for the rest of the thesis expressing algorithms in

terms of Turing machines would be too complicated and would obscure the intuition behind them. As a consequence we declare ourselves proponents of the Church-Turing thesis and will describe all our algorithms in natural language.

Definition 32. *A language $L \subseteq \Sigma^*$ is said to be* recursive *or* decidable[2] *if there is an algorithm which, given $w \in \Sigma^*$, outputs "yes" if $w \in L$ and outputs "no" if $w \notin L$.*

$L \subseteq \Sigma^*$ is said to be recursively enumerable *if there is an algorithm which, given $w \in \Sigma^*$, outputs "yes" if and only if $w \in L$ (but which may not terminate otherwise).*

It is clear that a subset of Σ^* is recursive if and only if both it and its complement in Σ^* are recursively enumerable.

The class of recursive and recursively enumerable languages will be denoted by $\mathcal{R}$ and $\mathcal{RE}$ respectively. We have the following strict inclusions:

$$\mathcal{REG} \subset \mathcal{OC} \subset \mathcal{CF} \subset \mathcal{R} \subset \mathcal{RE}.$$

We have already seen witnesses for the strictness of the first two inclusions. A very famous example for the strictness of the last one is the halting set: the set of program/input pairs such that the given program halts on the given input is recursively enumerable but not recursive. To see these pairs as a formal language all that is needed is to encode the pairs into a specific alphabet. We only need a one-letter alphabet to find an example of a recursive language which is not context-free: the language $L = \{a^p : p \text{ is prime}\}$ is well known to be recursive, as we have an algorithm to determine whether a number is prime or not. However it can be shown not to be context free: one possible method is to use the pumping lemma for context-free languages, which we will not state here. The intuitive explanation is that if this set

[2]One sometimes also says *solvable* about a language which is the word problem of a group.

were context-free then the sequence of primes would have an arithmetic subsequence.

3.2 Properties of classes of formal languages

We will list here the properties of the five classes of languages mentioned. We describe which properties are often grouped together and given a special name, and also which are consequences of other properties. For more details and proofs, see pages 270-284 of [26].

Definition 33. *A class $\mathcal{F}$ of languages is said to be a* cone *or* full trio *if:*

1. *It is* closed under homomorphism*, i.e. if $L \subseteq \Sigma^*$ is an element of $\mathcal{F}$ and $\psi : \Sigma^* \to \Omega^*$ is a monoid homomorphism, then $\psi(L) \in \mathcal{F}$.*
2. *It is* closed under inverse homomorphism*, i.e. if $L \subseteq \Omega^*$ is an element of $\mathcal{F}$ and $\psi : \Sigma^* \to \Omega^*$ is a monoid homomorphism, then $\psi^{-1}(L) \in \mathcal{F}$.*
3. *It is* closed under intersection with regular languages*, i.e. if when $L \subseteq \Sigma^*$ is in $\mathcal{F}$ and $R \subseteq \Sigma^*$ is a regular language, then $L \cap R \in \mathcal{F}$.*

Definition 34. *A class of languages is called a* trio *if it is closed under inverse homomorphism, intersection with regular languages, and λ-free homomorphism, meaning homomorphisms where no letter is sent to the empty word λ.*

Definition 35. *A class of languages is called a* (full) semi-AFL *if it is a (full) trio and is furthermore closed under union.*

Definition 36. *A class $\mathcal{F}$ of languages is called a* (full) AFL *if it is a (full) semi-AFL and furthermore closed under*

1. *Concatenation: If $L, K \in \mathcal{F}$, then so is $LK := \{lk : l \in L, k \in K\}$.*

2. *Kleene star: If $L \in \mathcal{F}$, then so is $L^* := \{\lambda\} \cup \bigcup_{n=1}^{\infty} L^n$ (where L^n is L concatenated with itself n times).*

There are a few other properties that are worth mentioning here and that will be used at some point in this thesis.

Definition 37. *A* generalized state machine *or* GSM $\mathfrak{M}$ *is a finite state automaton $(Q, \Sigma, \Delta, \delta, s, F)$ with an output alphabet Δ such that each transition can have an output word. More specifically, δ is a function from $Q \times \Sigma$ to finite subsets of $Q \times \Delta^*$, where $(p, w) \in \delta(q, a)$ is taken to mean that if the automaton is in state q reading a, it may chose to move to state p while outputting w. We then extend δ to $Q \times \Sigma^*$ inductively as expected:*

- *$\delta(q, \lambda) = \{(q, \lambda)\}$ for all $q \in Q$, and*
- *For each $x \in \Sigma^*$ and $a \in \Sigma$, $(p, w) \in \delta(q, xa)$ if and only if there are $w_1, w_2 \in \Omega^*$ and a state $r \in Q$ such that $w = w_1 w_2$, $(r, w_1) \in \delta(q, x)$ and $(p, w_2) \in \delta(r, w_1)$.*

Thus for a word accepted by the GSM, the output word is the concatenation of all outputs of its transitions. So the GSM M defines a mapping $\theta_{\mathfrak{M}} : \Sigma^* \to \Delta^*$, which sends a language L to the language

$$\{y \in \Delta^* : (p, y) \in \delta(s, x) \text{ for some } x \in L, p \in F\}$$

and which is called a GSM mapping.

The function $\theta_{\mathfrak{M}}^{-1} : \Delta^* \to \Sigma^*$ sending a language $L \subseteq \Delta^*$ to

$$\{x \in \Sigma^* : (p, y) \in \delta(s, x) \text{ for some } y \in L, p \in F\}$$

is in turn called an inverse GSM mapping *(note that this is not a true functional inverse).*

A GSM mapping is said to be λ-free if no transition has empty output.

Any monoid homomorphism can be seen as a GSM mapping. Also, for any language L and regular language R, one can define a GSM mapping sending L to $L \cap R$. On the other hand, any GSM mapping can be expressed as a combination of homomorphisms, inverse homomorphisms and intersections with regular languages, and any trio is closed under inverse GSM mappings (see [26] Theorem 11.1 and 11.2). Hence:

Proposition 38. *A class of languages is closed under GSM mappings and inverse GSM mappings if and only if it is a cone.*

Lastly, we define substitutions:

Definition 39. *A* substitution *is a monoid homomorphism from Σ^* for an alphabet Σ to the monoid $\wp(\Delta^*)$ consisting of all languages over that alphabet, under the concatenation operation. Thus, for a language $L \subseteq \Sigma^*$ it substitutes a letter of a word in L by another language $K \subseteq \Delta^*$.*

A substitution is said to be a regular substitution *if all languages replacing letters of the source words are required to be regular.*

Proposition 40. *$\mathcal{REG}, \mathcal{OC}, \mathcal{CF}$ and $\mathcal{RE}$ are full AFLs. $\mathcal{R}$ is an AFL only, as it is not closed under arbitrary homomorphisms.*

Table 3.1 keeps track of which formal language class has which properties. We include in it the class of terminal Petri net languages ($\mathcal{PNL}$), which will only be introduced in Chapter 6, for reference. By 'substitution' we mean substitution of languages of a class into languages of the same class.

Table 3.1: Closure properties of language classes

Property	$\mathcal{REG}$	$\mathcal{OC}$	$\mathcal{CF}$	$\mathcal{PNL}$	$\mathcal{R}$	$\mathcal{RE}$
$^{-1}$ homomorphism	✓	✓	✓	✓	✓	✓
λ-free homomorphism	✓	✓	✓	✓	✓	✓
homomorphism	✓	✓	✓	×	×	✓
$\cap\mathcal{REG}$	✓	✓	✓	✓	✓	✓
λ-free GSM	✓	✓	✓	✓	✓	✓
GSM	✓	✓	✓	×	×	✓
$^{-1}$ GSM	✓	✓	✓	✓	✓	✓
Union	✓	✓	✓	✓	✓	✓
Concatenation	✓	✓	✓	✓	✓	✓
Kleene *	✓	✓	✓	×	✓	✓
Intersection	✓	×	×	✓	✓	✓
Complement	✓	×	×	×	✓	×
Regular substitution	✓	✓	✓	λ-free	✓	✓
Self-substitution	✓	×	✓	×	×	✓
λ-free substitution	✓	×	✓	reg	✓	✓

3.3 Word problems of groups and their language classes

The aforementioned properties of our five classes of languages will have important consequences in terms of groups having word problems in those classes. The most important among these is closure under inverse homomorphism. This property of a language class $\mathcal{F}$ ensures that even though the word problems with respect to two generating sets can differ vastly, they cannot differ so much that one lies in the class $\mathcal{F}$ and the other does not.

The following is a well-known result (see [21] and [22] for example):

Theorem 41. *Let G be a finitely generated group and $\phi : (A \cup A^{-1})^* \to G$ and $\psi : (B \cup B^{-1})^* \to G$ two surjective monoid homomorphisms. Then there is a monoid homomorphism $\theta : (A \cup A^{-1})^* \to (B \cup B^{-1})^*$ such that $\psi \circ \theta = \phi$.*

This essentially means that for a class of languages closed under inverse

homomorphisms, translating into another alphabet does not take us out of the class. This gives us a very nice property (see [22] for example):

Proposition 42. *If a class of languages $\mathcal{F}$ is closed under inverse homomorphism and the word problem of a group G with respect to some finite generating set lies in $\mathcal{F}$ then the word problem of G with respect to any finite generating set lies in $\mathcal{F}$.*

This now means that for any class $\mathcal{F}$ which is closed under inverse homomorphisms and any finitely generated group G we can talk about "the" word problem of G being in $\mathcal{F}$ without any reference to the choice of generating set. In this case we will (mildly) abuse notation and write $G \in \mathcal{F}$.

Definition 43. *Let $\mathcal{F}$ be a class of languages closed under inverse homomorphisms, G a finitely generated group. The set of $\mathcal{F}$-subsets of G is*

$$\mathcal{F}(G) := \{S \subseteq G : \exists \text{ surjective monoid homomorphism } \phi \text{ s.t. } \phi^{-1}(S) \in \mathcal{F}\}.$$

So $G \in \mathcal{F}$ means that $\{1_G\} \in \mathcal{F}(G)$.

The other properties of a cone also help preserve membership in a cone when performing algebraic operations on groups. All the classes mentioned before thus satisfy the following two theorems. In their statements, we specify only the minimum conditions on language families for them to hold.

Theorem 44. *Let $\mathcal{F}$ be a class of languages closed under inverse homomorphism and intersection with regular languages. Let $G \in \mathcal{F}$, and let H be a finitely generated subgroup of G. Then $H \in \mathcal{F}$ as well.*

Proof. See Lemma 2 in [25]: if H is finitely generated, it has a finite generating set A. Choose a generating set B for G that includes A. Then the word problem of H is just

$$W_A(H) = W_B(G) \cap (A \cup A^{-1})^*$$

which is in $\mathcal{F}$ since $(A \cup A^{-1})^*$ is regular. □

Note that arbitrary subgroups of finitely generated groups are not necessarily finitely generated. However, finite index subgroups are. So the above theorem holds in particular for finite index subgroups H. How about the opposite? If we know that H is in $\mathcal{F}$, when can we say that $G \in \mathcal{F}$?

Theorem 45. *Let $\mathcal{F}$ be a class of languages closed under union with regular sets and inverse GSM mappings. Let H be a finitely generated group and let G be such that $[G : H] < \infty$. Then $G \in \mathcal{F}$.*

Proof. See Lemma 5 in [25]. The idea is that we can write a GSM mapping which essentially keeps track of which coset of H in G we are in, and translates a word in a coset Hg to a word wg where w represents a word in H. □

Apart from closure properties of classes of groups with word problem in a certain language family, we also have a classification of the algebraic structure of such groups. The properties of groups whose word problem lies in a certain class of languages have been studied for a long time. One of the first correspondences in this direction is Anisimov's in [1]. We only sketch the proofs below.

Theorem 46. *A group is finite if and only if its word problem is a regular language.*

Proof Sketch: First done in [1]; see [43] for a simple proof. The idea for the left to right direction is to use the Cayley graph of the group as the finite automaton, with the unit being the start and accept state. □

Theorem 47. *A group is virtually free if and only if its word problem is a context-free language.*

Proof Sketch: See [43]. In fact, Muller and Schupp proved only that a group is virtually free if and only if it is context-free and accessible (see [43] itself for the definition of accessibility). Dunwoody proved two years later in [9] that any finitely presented group is accessible. It was already known that

context-free groups are finitely presented (see [2]), so we have the result in its current form. Muller and Schupp's proof uses the theory of ends and in particular Stallings' theorem on ends of groups (see Theorem 80) in an essential way. □

Theorem 48. *A group is virtually cyclic if and only if its word problem is a one-counter language.*

Proof. See [21]. Using the fact that one-counter groups are context-free, and the previous theorem, Herbst proved that the normal free subgroup of finite index in a one-counter group (we can take the free subgroup of finite index in a virtually free group to be normal) must have a single generator, because the Dyck language on two sets of parenthesis generates $\mathcal{CF}$ as a cone. The converse is more involved and follows from considerations about rational subsets, and from a result we state in Chapter 4 in the proof of Theorem 54. An intuitive explanation of why virtually cyclic groups are one-counter is given below. □

The intuition behind the previous theorems is as follows: with a regular language, one is dealing with a NFA, a machine with no memory. Thus after a certain length of word, one can no longer distinguish between group elements representing the identity and other elements (in an argument similar to that of the pumping lemma for regular languages). Only finitely many group elements are therefore representable in this way.

When a group is virtually cyclic, we can see each group element as having a cyclic component and a finite component indicating which coset of the cyclic subgroup it is in. In other words a group element has a normal form $z^k g$ where z is the element of infinite order generating the finite index cyclic subgroup, and g is a coset representative. Intuitively, we use the stack to keep track of the power of z, and the states of the one-counter automaton to keep track of the coset representatives. Similarly, for a virtually free group, we keep track of the cosets in the states of the pushdown automaton, but

since we now have various stack symbols, we can express elements of the free group in the stack: we can test whether a word represents the identity in a free group by writing down one symbol at a time and performing free reductions whenever possible; the word represents the identity if and only if we end up with the empty word, and this procedure can be implemented in a natural way using a stack.

Herbst [21] also showed that, if $\mathcal{F}$ is a cone that is a subset of the context-free languages, then the class of groups whose word problem lies in $\mathcal{F}$ is either the class of groups with a regular word problem, the class of groups with a one-counter word problem or the class of groups with a context-free word problem. There are therefore no more classes of groups corresponding to cones below the context-free languages.

Chapter 4

Pairs of groups and their ends

4.1 Definitions and motivation

Our aim here is to look at a generalization of the word problem: instead of asking which words over the generators of a group represent the identity, we ask which words represent a given subgroup. This is sometimes called the *membership problem*, but we will call it the word problem of the pair of groups. There are many versions of the membership problem - some have studied it with respect to rational subsets, some study the generalized membership problem, where the subgroup is not specified, and the question asks if the problem is decidable for every finitely generated subgroup. Some work in these areas has been done by Markus Lohrey, Benjamin Steinberg, Zeph Grunschlag, Mark Kambites, Pedro Silva and Claas Röver, amongst others.

We mostly follow the terminology used in [5].

Definition 49. *Let G be a finitely generated group, $K \leqslant G$ a subgroup. Let $\phi : \Sigma^* \to G$ be a finite group presentation. The* word problem of the the pair (G, K) with respect to ϕ *is the language*

$$L(G, K, \phi) := \{w \in \Sigma^* | \phi(w) \in K\}.$$

The language above clearly depends on the generating set we are taking for G. As with single groups, we will use the generating set and presentation interchangeably, sometimes writing $L(G, K, A)$ for a finite generating set A, or even $L(G, K, \Sigma)$ where $\Sigma = A \cup A^{-1}$. As with ordinary word problems, closure of the language class under inverse homomorphism is enough to make the class of the word problem of a pair independent of choice of generating set (the proof is also much the same here):

Lemma 50. *Let $\mathcal{F}$ be a class of languages closed under inverse homomorphism. If $L(G, K, \phi) \in \mathcal{F}$ for some finite presentation ϕ, then $L(G, K, \psi)$ is in $\mathcal{F}$ for every finite presentation ψ of G.*

Proof. This is a reformulation of the discussion around Lemma 2.1 in [22].

Let $\phi : \Sigma^* \to G$ and $\psi : \Omega^* \to G$ be the two presentations. It is well known that since these are two surjective monoid homomorphisms, then there is a "translation", a monoid homomorphism $\theta : \Omega^* \to \Sigma^*$ such that $\phi \circ \theta = \psi$. Now $\theta^{-1}(L(G, K, \phi)) = L(G, K, \psi)$, and because $\mathcal{F}$ is closed under inverse homomorphism and $L(G, K, \phi) \in \mathcal{F}$, $L(G, K, \psi)$ is also in $\mathcal{F}$. □

We can now simply talk about the pair (G, K) being in $\mathcal{F}$ when $\mathcal{F}$ is a class of languages closed under inverse monoid homomorphisms. Recall that all three language classes mentioned in this chapter (the class $\mathcal{REG}$ of regular languages, the class $\mathcal{CF}$ of context-free languages, and the class $\mathcal{OC}$ of one-counter languages) are cones and hence closed under inverse homomorphism.

In this chapter we want to learn about what the language class of the word problem of a pair of groups can tell us about the structure of the pair. For this, we study Schreier graphs, the analogue to Cayley graphs for pairs of groups, and the operations on groups which preserve membership of a pair in a class of languages.

Another question we try to answer is the relation between (G, K) being in a class $\mathcal{F}$ of languages and (G, K_G) being in $\mathcal{F}$, where K_G is the normal core of K in G (i.e. K_G is the largest normal subgroup of G contained in K

- see Definition 73). This is because the normal core is closely related to G being realized as a syntactic monoid (see Section 4.5 for further details).

We will first give some general facts about pairs of groups. Some results are gathered from previous work of T. Herbst, D. Holt, M. Owens and R. Thomas. Apart from these, our main inspiration is the paper [5], which gathers many of these results in the specific case of context-free groups. Many of them can be generalized from context-free languages to arbitrary cones without much trouble. Where more specific results are true for context-free and one-counter pairs, we will describe them.

We will then review some of the work Claas Röver has done in [53] - he looks at which groups can be syntactic monoids of languages in a certain class.

In an effort to use the theory of ends to help us answer the above questions, we also make a brief foray into this theory which was introduced in Chapter 2, where an invariant of the geometry of a pair of groups can tell us something about their algebraic relationship. End theory was mainly pioneered by Freudenthal and Hopf, and later Stallings proved a very important result. Much work on ends of pairs of groups has been done by P. Scott and G.A. Swarup.

4.2 General facts, regular pairs

4.2.1 Schreier graphs and their properties

When we are talking about pairs of groups, the graph to be considered is no longer the Cayley graph but a generalization: the Schreier graph of a group with respect to a subgroup.

Definition 51. *Let G be a group and A a finite generating set for G, ϕ its presentation map. Let K be a subgroup of G. The* Schreier graph $\Gamma(G, K, A)$ *of the pair (G, K) over A is a labelled oriented graph where the set of nodes*

is the set of cosets of K in G, and $Kg \xrightarrow{a} Kh$ for $a \in A \cup A^{-1}$ if and only if $Kg\phi(a) = Kh$.

This agrees with the definition of the Cayley graph in a natural way: if K is the trivial subgroup, then the Schreier graph is just the Cayley graph of G. Furthermore, if K is normal in G, then the Schreier graph of (G, K) is in fact isomorphic to the Cayley graph of the quotient group G/K (over the same generating set[1]). In fact, we have an even more general result, which is folklore but which we spell out here:

Proposition 52. *Let G be a finitely generated group, Σ a set closed under inversion generating it, and ϕ its presentation map. Let $K \leqslant G$, and let $N \leqslant K$ be a normal subgroup of G. Then the Schreier graph $\Gamma(G/N, K/N, \Sigma)$ is isomorphic to the Schreier graph $\Gamma(G, K, \Sigma)$.*

Proof. The map $\psi : \Gamma(G, K, \Sigma) \to \Gamma(G/N, K/N, \Sigma)$ defined by

$$Kg \mapsto (K/N)(Ng)$$

is easily shown to be a labelled graph isomorphism.

- It is well-defined and one-to-one: for all $x, y \in G$,

$$\begin{aligned} Kx = Ky &\iff xy^{-1} \in K \\ &\iff Nxy^{-1} \in K/N \\ &\iff NxNy^{-1} = Nx(Ny)^{-1} \in K/N \text{ because } N \text{ is normal} \\ &\iff (K/N)(Nx) = (K/N)(Ny). \end{aligned}$$

- It is clearly onto.

[1]Technically, if $\{a_1, \ldots, a_n\}$ is the generating set for G, then we are taking the generating set $\{Ka_1, \ldots, Ka_n\}$ for G/K.

- It is a labelled graph homomorphism: assume that $Kx \xrightarrow{a} Ky$ for a generator a of G. Here we will identify a with its image $\phi(a)$ in G, to ease clutter. We will show that $(K/N)(Nx) \xrightarrow{Na} (K/N)(Ny)$.

$$\begin{aligned} Kx \xrightarrow{a} Ky &\iff Kxa = Ky \\ &\iff xay^{-1} \in K \\ &\iff Nxay^{-1} \in K/N \\ &\iff (Nx)(Na)(Ny)^{-1} \in K/N \text{ because } N \text{ is normal} \\ &\iff (K/N)(Nx)(Na) = (K/N)(Ny) \\ &\iff (K/N)(Nx) \xrightarrow{Na} (K/N)(Ny). \end{aligned}$$

□

There are more analogies to be made with the Cayley graph. For nodes $x, y \in \Gamma(G, K, \Sigma)$ define the language

$$L_{x,y} := \{w \in \Sigma^* : x\phi(w) = y\}.$$

This is the language of labels of paths in the Schreier graph leading from node x to node y.

We can take the origin in the Schreier graph $\Gamma(G, K, \Sigma)$ to be the coset K. Just as in a Cayley graph, it is obvious that $L(G, K, \Sigma)$ is just $L_{K,K}$. What can we say about $L_{x,y}$ for other nodes in the Schreier graph? First we will need the following definition:

Definition 53. *Let G be a finitely generated group. For any $S \subseteq G$, we define the set of* rational subsets of G *as follows: $S \in \mathcal{RAT}(G)$ if and only if*

$$\exists \phi \text{ onto monoid homomorphism, } \exists L \text{ regular s.t. } \phi(L) = S.$$

Note that there are alternative definitions of rational subsets (see [21] for example) but they are equivalent to this one in the case of finitely generated

groups.

The following result was proven in [5] for context-free languages but we give a different proof for general families of languages:

Theorem 54. *If $\mathcal{F}$ is closed under concatenation with regular languages, and $(G, K) \in \mathcal{F}$, then $L_{K,Kg} \in \mathcal{F}$ for all $g \in G$.*

Proof. For every $g \in G$, $\{g\} \in \mathcal{RAT}(G)$: just take a singleton consisting of a word representing g. The image of this set under ϕ is $\{g\}$, and it is a finite and hence regular set.

In [21], it is proven that for any language family $\mathcal{F}$ closed under concatenation with regular languages, if $R \in \mathcal{RAT}(G)$ and $T \in \mathcal{F}(G)$, then $RT, TR \in \mathcal{F}(G)$. We know by assumption that $K \in \mathcal{F}(G)$. Hence for any $g \in G$, $Kg \in \mathcal{F}(G)$. But this means exactly that $L_{K,Kg} \in \mathcal{F}$. □

As a corollary of the proof, since we can concatenate with rational subsets both on the right and on the left, we have the following:

Corollary 55. *If $\mathcal{F}$ is a family closed under concatenation with regular languages and $(G, K) \in \mathcal{F}$, then $(G, g^{-1}Kg) \in \mathcal{F}$ for any $g \in G$.*

We shall see below that whether or not $L(G, K, A)$ lies in a particular cone does not change when we vary the generating set. We have a similar result for Schreier graphs; essentially, the Schreier graphs for two generating sets cannot be too different. Again, we have the more general result stated in Theorem 58 below.

Recall the notion of distance in a Cayley graph, defined in Chapter 2. We use the same metric for Schreier graphs. We now define quasi-isometry, which tells us when two graphs have the same "coarse geometry", i.e. whether they look the same when squinted at from a distance. The following definition is stated in terms of metric spaces.

Definition 56. *Let (X, d_X) and (Y, d_Y) be two metric spaces, $f : X \to Y$ a function. We say that f is a* quasi-isometry *if*

1. *There is a constant* $\alpha \geqslant 1$ *such that for all* $x, x' \in X$,

$$\frac{1}{\alpha} d_X(x, x') - \alpha \leqslant d_Y(f(x), f(x')) \leqslant \alpha d_X(x, x') + \alpha.$$

2. *There is a constant* α *such that for all* $y \in Y$ *there is* $x \in X$ *such that*

$$d_Y(f(x), y) \leqslant \alpha.$$

The smallest constant α *satisfying both conditions above is called the* quasi-isometry constant *of* f.

It is fairly straightforward to prove the following:

Theorem 57. *Quasi-isometry is an equivalence relation over metric spaces.*

Proof. See Proposition 11.39 in [41]. □

Theorem 58. *Let* G *be a finitely generated group, and* $K \leqslant H \leqslant G$ *such that* $[G : H] < \infty$. *Let* S *(closed under inversion) be a generating set for* H. *Let* T *be a set of coset representatives of* H *in* G, *excluding the representative for* H. *Then* $S' = S \cup T \cup T^{-1}$ *is a group generating set for* G.

Then the Schreier graphs $\Gamma(H, K, S)$ *and* $\Gamma(G, K, S')$ *are quasi-isometric.*

Proof. To simplify notation, we will set $\Gamma := \Gamma(H, K, S)$ and $\overline{\Gamma} := \Gamma(G, K, S')$. First note that with this setting of generating sets, Γ is a subgraph of $\overline{\Gamma}$. The quasi-isometry is therefore just the inclusion function, and we just need to show the following two properties:

1. $\exists \alpha \geqslant 1$ s.t. $\forall Kh, Kh' \in H/K$,

$$\frac{1}{\alpha} d_\Gamma(Kh, Kh') - \alpha \leqslant d_{\overline{\Gamma}}(Kh, Kh') \leqslant \alpha d_\Gamma(Kh, Kh') + \alpha.$$

2. $\exists \alpha$ s.t. $\forall Kg \in G/K$,$\exists Kh \in H/K$

$$d_{\overline{\Gamma}}(Kh, Kg) \leqslant \alpha.$$

Note that in this case, H/K just denotes the set of cosets of K in H, as K is not necessarily normal.

Since Γ is a subgraph of $\overline{\Gamma}$, we have that $d_{\overline{\Gamma}}(Kh, Kh') \leqslant d_{\Gamma}(Kh, Kh')$ for all $Kh, Kh' \in H/K$, and we do not need to prove the second inequality of property (1).

Let us prove (2). Let $Kg \in G/K$ be any coset of K in G. g is in a certain coset of H in G, say $g \in Ht$ for $t \in T$. Then there is $h \in H$ such that $g = ht$. But then $Kh \xrightarrow{t} Kg$, and therefore $d_{\overline{\Gamma}}(Kh, Kg) \leqslant 1$. We can therefore take $\alpha = 1$.

Now for the first inequality of (1). Let $Kh, Kh' \in H/K$. Look at a $\overline{\Gamma}$-path π of shortest length from Kh to Kh'. Some of the nodes on this path are in H/K, some are not. Look at consecutive nodes Kh_i, Kh_{i+1} in H/K on π, where by "consecutive" we mean that any node (if any such node exists) between Kh_i and Kh_{i+1} on the path π is in $(G/K) - (H/K)$. Look at the following sets:

$$
\begin{aligned}
&S,\\
&\{st^{-1} : s \in S', t \in T\} \cap H,\\
&\{t_1 s t_2^{-1} : t_1, t_2 \in T, s \in S'\} \cap H,\\
&\{ts : s \in S', t \in T\} \cap H.
\end{aligned}
$$

These are all finite sets, all of whose elements are in H, so there is an upper bound to their shortest expression as words over S (in other words, their expression in terms of generators of H and their inverses). Call this upper bound N. Now, for each stretch of the path π between consecutive H/K-nodes Kh_i and Kh_{i+1}, one of the following holds:

1. Kh_i and Kh_{i+1} are really adjacent on π, i.e. $Kh_i \xrightarrow{a} Kh_{i+1}$ is a portion of π for $a \in S'$, or

2. There is at least one node in $(G/K) - (H/K)$ between Kh_i and Kh_{i+1} on π.

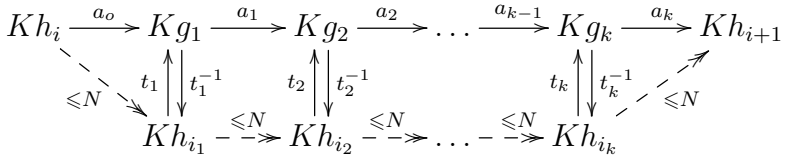
In the first case, since $Kh_i a = Kh_{i+1}$, $h_i a h_{i+1}^{-1} \in K \subseteq H$.
Therefore $a \in H \cap S' = S$ and this means that

$$d_\Gamma(Kh_i, Kh_{i+1}) = d_{\overline{\Gamma}}(Kh_i, Kh_{i+1}) = 1.$$

In the second case, suppose the concerned portion of π is

$$Kh_i \xrightarrow{a_0} Kg_1 \xrightarrow{a_1} Kg_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{k-1}} Kg_k \xrightarrow{a_k} Kh_{i+1}$$

where $Kg_1, \ldots, Kg_k \in (G/K) - (H/K)$, $a_0, \ldots, a_k \in S'$. Recall that for each Kg_j where $1 \leqslant j \leqslant k$ there is a Kh_{i_j} at distance one from it. In fact, there is t_j in T such that $Kh_{i_j} \xrightarrow{t_j} Kg_j$ and $Kg_j \xrightarrow{t_j^{-1}} Kh_{i_j}$. So we have the diagram below:



Since all of $a_0 t_1^{-1}, t_1 a_1 t_2^{-1}, \ldots, t_k a_k$ are in the sets mentioned above, there are paths purely in Γ between Kh_i and Kh_{i_1}, Kh_{i_1} and Kh_{i_2} and so on, whose lengths (in Γ) do not exceed N. So we have that

$$d_\Gamma(Kh_i, Kh_{i+1}) \leqslant N d_{\overline{\Gamma}}(Kh_i, Kh_{i+1})$$

in both cases.

Denote the nodes of H/K along π by $Kh, Kh_1, Kh_2, \ldots, Kh_n, Kh'$. Then

$$\begin{aligned} d_\Gamma(Kh, Kh') &\leqslant d_\Gamma(Kh, Kh_1) + \ldots + d_\Gamma(Kh_n, Kh') \\ &\leqslant N\left(d_{\overline{\Gamma}}(Kh, Kh_1) + \ldots + d_{\overline{\Gamma}}(Kh_n, Kh')\right) \\ &= N d_{\overline{\Gamma}}(Kh, Kh'), \end{aligned}$$

because π is a shortest path in $\overline{\Gamma}$ and because the triangular inequality always holds in a graph. We can now take $\alpha = max\{1, N\}$ and we are done. □

Of course, taking $G = H$ this shows that:

Corollary 59. *The Schreier graphs of (G, K) with respect to two different presentations/generating sets are always quasi-isometric. In particular, the Cayley graphs of a group with respect to different generating sets are also quasi-isometric.*

4.2.2 Certain properties regarding cones and pairs

Now, there are a certain number of theorems we can state about word problems of pairs of groups which reside in a class of languages. Again, a proof of the following theorem is given in Lemma 3.1 of [5], but we give a more general and language-theoretic proof, whereas their proof is automata-theoretic and is specific to context-free languages.

Theorem 60. *Let $\mathcal{F}$ be family of languages closed under inverse homomorphism, G a finitely generated group and H a finitely generated subgroup of G. Let $\phi : \Sigma^* \to G$ be a finite presentation of G and $\psi : \Omega^* \to H$ a finite presentation of H. Then for any subgroup K of G, if $L := L(G, K, \phi) \in \mathcal{F}$ then $L' := L(H, K \cap H, \psi) \in \mathcal{F}$.*

Proof. We have essentially the same proof as in Lemma 50 above. The diagram below commutes; $\psi \circ \iota = \theta \circ \phi$ on Ω^* (where ι is just the inclusion

mapping). It is therefore straightforward to show that $L' = \theta^{-1}(L)$ and we are done.

$$\begin{array}{ccc} \Sigma^* & \xrightarrow{\phi} & G \\ \theta \uparrow & & \uparrow \iota \\ \Omega^* & \xrightarrow{\psi} & H \end{array}$$

□

Clearly, taking $H = G$ in the theorem above, we get back Lemma 50. Recall the similar result about Schreier graphs in the previous section - the two are not unrelated. Intuitively, there is a bound to how much longer paths can get in a Schreier graph with respect to another generating set. If one thinks of the typical examples of cones - regular, context-free, one-counter (corresponding to finite automata, pushdown automata and one-counter automata respectively), this is nothing we cannot handle in the states of the automaton, so we remain in the same class of languages.

Furthermore, keeping $H \leqslant G$ and taking $K \leqslant H$, we also get that the class of languages a word problem of a pair is in is closed under taking finite index subgroups of the larger group:

Corollary 61. *If $\mathcal{F}$ is closed under inverse homomorphism, $K \leqslant H \leqslant G$ and H is finitely generated, then*

$$(G, K) \in \mathcal{F} \Rightarrow (H, K) \in \mathcal{F}.$$

The converse of this corollary is true when $\mathcal{F}$ is closed under inverse GSM mappings and H has finite index in G:

Theorem 62. *Let $\mathcal{F}$ be a family of languages closed under inverse GSM mappings. Let G be a finitely generated group, and K and H two subgroups such that $K \leqslant H$ and $[G : H] < \infty$. Then*

$$(H, K) \in \mathcal{F} \Rightarrow (G, K) \in \mathcal{F}.$$

Proof. This proof is taken from [25]. Since H has finite index in G, it is also finitely generated. Call its set of generators (together with their inverses) Σ. Let T be a set of coset representatives of H in G, including 1. Complete Σ to a generating set Ω for G by adding the set of representatives: $\Omega := \Sigma \cup (T - \{1\})$.

For each pair $t \in T$ and $y \in \Omega^*$, $ty \in Ht'$ for some $t' \in T$; pick an element $h_{ty} \in \Sigma^*$ such that $ty =_G h_{ty}t'$. Now, the generalized sequential machine mapping $\gamma : \Omega^* \to \Sigma^* T$ which takes every such ty to the corresponding $h_{ty}t'$ clearly takes any word w over Ω to a word $w't'$ indicating the coset of H w is in. In other words, $w' \in \Sigma^*$ and $w =_G w't'$.

We now want to show that $(G, K) = \gamma^{-1}(H, K)$. Let $w \in (H, K)$ be an element whose γ-image is $w' \in (H, K)$. Then necessarily $w' \in \Sigma^* \subset \Omega^*$ (as w must represent an element of K, which is in H). Since γ preserves the element of G represented, $w' \in (G, K)$. Conversely, if $w \in (G, K)$ then $\gamma(w) \in (H, K)$ as again, the coset representative must be unity.

Since $(H, K) \in \mathcal{F}$, so is $(G, K) = \gamma^{-1}(H, K)$ since $\mathcal{F}$ is closed under inverse GSM mappings. □

Note that the classes of regular, one-counter and context-free languages are all closed under inverse GSM mappings.

There are also some manipulations we are allowed to do to the smaller group without changing the class of languages:

Theorem 63. *Let $\mathcal{F}$ be a family of languages closed under finite unions and concatenation with regular languages. Say K has finite index in $H \leqslant G$. Then*

$$(G, K) \in \mathcal{F} \Rightarrow (G, H) \in \mathcal{F}.$$

Proof. The idea of the proof is from [5] Lemma 4.11. Look at the Schreier graph $\Gamma := \Gamma(G, K, \phi)$ for an arbitrary finite presentation ϕ. Take the finite set H/K of cosets of K in H, and look at the nodes Kh corresponding to these in Γ. Then $L(G, H, \phi) = \bigcup_{x \in H/K} L_{K,x}$ is a finite union of languages in $\mathcal{F}$ (according to Theorem 54), so we are done. □

Thus the language class a word problem is in is closed under taking finite extensions of the smaller group. Note that if we take K to be the trivial subgroup then we get the following:

Corollary 64. *If the word problem of G is in a language family $\mathcal{F}$ closed under finite unions, and if K is a finite subgroup of G, then $(G, K) \in \mathcal{F}$.*

A remark that should be made here is that the converse to Theorem 63 does not hold: in Example 5.9 of [5] such a pair of subgroups is constructed for the family of context-free languages.

We can also quotient out normal subgroups:

Theorem 65. *Let $\mathcal{F}$ be a family closed under inverse homomorphisms. Let $K \leqslant G$ and let N be a normal subgroup of G contained in K. Then*

$$(G, K) \in \mathcal{F} \iff (G/N, K/N) \in \mathcal{F}.$$

Proof. Let $\pi : G \to G/N$ be the natural projection, and let ϕ be the presentation mapping for G.

$(\Rightarrow)$: $\phi \circ \pi$ is a monoid homomorphism giving a finite presentation of the quotient G/N. Furthermore, $(\phi \circ \pi)^{-1}(K/N) = \phi^{-1}(K)$. So $(G/N, K/N) \in \mathcal{F}$.

$(\Leftarrow)$: By assumption, there is a language in $\mathcal{F}$ mapping onto K/N via a presentation ψ of G/N. But $\phi \circ \pi$ is also a presentation of G/N, and since $\mathcal{F}$ is closed under inverse homomorphism, $\phi^{-1}(K) = (\phi \circ \pi)^{-1}(K/N) \in \mathcal{F}$. But this is exactly (G, K). □

Finally, we have an expected result for regular languages: a pair is regular if and only if the small group has finite index in the larger one. The proof is essentially the same as Anisimov's proof for the word problem [1].

Theorem 66. *Let G be a finitely generated group and K a subgroup. Then*

$$(G, K) \in \mathcal{REG} \iff [G : K] < \infty.$$

Proof. See [5] for a simple proof, essentially using the Schreier graph to define the NFA. For a more algebraic proof, see [22]. □

4.3 Context-free pairs

As the class $\mathcal{CF}$ of context-free languages is a cone, and furthermore closed under inverse GSM mappings, finite unions and concatenation with regular sets, it satisfies all the properties listed in Section 4.2 above.

To go further, we will have to introduce some concepts relating to the geometry of the Schreier graph of a pair of groups. Recall the concept of balls of diameter n in a graph.

Definition 67. *Let Γ be a graph with designated origin o. Recall that for every natural number n*

$$B(o,n) = \{x \in V(\Gamma) | d_\Gamma(o,x) \leqslant n\}.$$

We say that Γ is a context-free graph with respect to F *if there are only finitely many isomorphism classes (as labelled graphs) of connected components of $\Gamma - B(o,n)$ across all ns.*

Now, it turns out that this notion of context-freeness of graphs coincides perfectly with our notion of a context-free language for pairs of groups. The proof can be found in [5] and is modelled after (and generalizes) Muller and Schupp's 1985 paper [42] where they prove the same thing for Cayley graphs rather than Screier graphs. More explicitly:

Theorem 68. *The Schreier graph $\Gamma(G,K,\phi)$ is context-free in the above sense if and only if the language $L(G,K,\phi)$ is context-free in the ordinary sense. Furthermore, in this case $L(G,K,\phi)$ is deterministic context-free.*

Proof. The left to right direction is contained in Theorem 4.2 and the right to left direction in Theorem 4.6 of [5]. □

We need an additional lemma to prove that any pair consisting of a virtually free group and its finitely generated subgroup is context-free (the overall proof is mainly taken from [5] but the proof of the Lemma below is a bit different):

Lemma 69. *Let F be a finitely generated free group, K a finitely generated subgroup. Then the pair (F, K) is context-free.*

Proof. Let A be a finite generating set for F. Note that K inherits freeness from G. From Remark 2.8 and Proposition 3.8 in [33], we know that the set of all freely reduced[2] words over $A \cup A^{-1}$ representing elements of K is a regular language - call it L.

Now, to get the full membership problem of K in F (not just the freely reduced words), we need to insert elements of $W_A(F)$ into L. In fact, the *parallel insertion* $(L \Leftarrow W_A(F))$ is enough. This operation is defined as:

$$(L_1 \Leftarrow L_2) = \bigcup_{u \in L_1} (u \Leftarrow L_2)$$

for $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Omega^*$ where

$$(u \Leftarrow L_2) := \{v_0 a_1 \dots a_k v_k : a_j \in \Sigma, v_i \in L_2, u = a_1 \dots a_k\}.$$

We know from Theorem 2 in [34] that any class of languages closed under concatenation and λ-free substitution is closed under parallel insertion. Since the context-free languages have these properties (see [26] for example), we have that

$$L(G, K, A) = (L \Leftarrow WP_A(F))$$

is context-free. □

[2] A word is said to be *freely reduced* if it has no subwords which would represent 1 in the free group over the same generators.

Theorem 70. *If H is a finitely generated subgroup of a finitely generated group G, then*

$$G \in \mathcal{CF} \Rightarrow (G, H) \in \mathcal{CF}.$$

Proof. If G is in $\mathcal{CF}$ then it is virtually free - let F be a free subgroup of finite index. Then $K := F \cap H$ is also a free group and it has finite index in H. So it also inherits finite generation from H. We know that $(F, K) \in \mathcal{CF}$ by Lemma 69. By Theorem 62, $(G, K) \in \mathcal{CF}$. By Theorem 63, so is (G, H) and we are done. □

Note that though context-free languages are closed under λ-free substitution, one-counter languages are not. The proof of Lemma 69 is therefore not valid in that case. However, due to the simpler structure of one-counter languages, the overall theorem still holds, for a different reason which we will see in the next section.

4.4 One-counter pairs

In the case of one-counter pairs, the larger group being one-counter gives us some information about the properties of an arbitrary subgroup. The following theorem is folklore; we give here a simple proof.

Lemma 71. *If $G \in \mathcal{OC}$ and $K \leqslant G$, then either $|K| \leqslant \infty$ or $[G : K] < \infty$.*

Proof. Recall from the previous chapter that if G is a one-counter group, then G is either finite or it has a finite index subgroup $H \simeq \mathbb{Z}$. If G is finite then we are done; so assume we have this infinite cyclic subgroup H.

We have $H \cap K \leqslant H$. If $H \cap K$ is non-trivial, then, since all non-trivial subgroups of $\mathbb{Z}$ are of the form $n\mathbb{Z}$ for some $n \in \mathbb{Z}$, we can deduce that $H \cap K$ has finite index in H. Now from the equation

$$[G : H \cap K] = [G : H][H : H \cap K]$$

we get that $[G : H \cap K]$ is finite. But since $[G : K] \leqslant [G : H \cap K]$, the former is also finite.

So we may assume that $H \cap K = \{1_G\}$. We now show that for any coset Hg of H in G, $|Hg \cap K| \leqslant 1$. Say $k, k' \in K$ are in the same coset, i.e. $k' \in Hk$. Then $k'k^{-1} \in H \cap K = \{1\}$, so that $k' = k$. Now $|K| \leqslant [G : H]$ and therefore K is finite. □

The property of being one-counter therefore transfers from the larger group G to the pair (G, K) for any K:

Lemma 72. *If G is one-counter and $K \leqslant G$, then (G, K) is a one-counter pair.*

Proof. By Lemma 71, either K or its index in G is finite. In the second case, (G, K) is a regular pair, hence also one-counter. So assume $|K| < \infty$. Then by Corollary 64, (G, K) is one-counter. □

4.5 Normal cores of subgroups and syntactic monoids

After having shown that the class in which a pair lies can be closed under taking finite index subgroups, finite extensions and so on, we will here take a closer look at a particular operation, namely taking the normal core of a subgroup in a larger group.

Our aim is to try to answer the following question for certain classes $\mathcal{F}$ of languages:

Question 1: If $(G, K) \in \mathcal{F}$, is it true that $(G, K_G) \in \mathcal{F}$?

The normal core of a subgroup in a group is defined as follows:

Definition 73. *Let G be a group, K a subgroup. The* normal core K_G *of K in G is the largest normal subgroup of G contained in K.*

It is well known that $K_G = \bigcap_{g \in G} g^{-1} K g$.

The reason Question 1 is an interesting question is that it is equivalent to one about syntactic monoids:

Question 2: If a language L is in $\mathcal{F}$, a group G its syntactic monoid, and $[L]$ a subgroup of G, when can we deduce that $G \in \mathcal{F}$?

Let us explain the details of the question above. The syntactic monoid of a language $L \subseteq \Sigma^*$ is the smallest monoid which is a homomorphic image of L and which distinguishes words in L from words outside of L. It is formally defined as follows:

Definition 74. *Let $L \subseteq \Sigma^*$. Then the* syntactic congruence *is the equivalence relation $\sim_L$ defined on Σ^* as follows:*

$$w \sim_L u \iff \forall x, y \in \Sigma^* (xwy \in L \iff xuy \in L)$$

for every $w, u \in \Sigma^*$. Then the syntactic monoid of L *is the monoid $\Sigma^* / \sim_L$.*

$[L]$ is then the image of L under the natural map $[\ \ ] : \Sigma^* \to \Sigma^* / \sim_L$ sending each word to its equivalence class.

Let us first establish that the two questions are equivalent:

Proposition 75. *The answer to Question 1 is positive if and only if the answer to Question 2 is positive.*

Proof. ($\Rightarrow$) : Let $L \subseteq \Sigma^*$ be a language in $\mathcal{F}$, let $G = \Sigma^* / \sim_L$, and suppose that $[L]$ is a subgroup of G. We will show that the only proper normal subgroup of G contained in $[L]$ is the trivial one. Then a positive answer to Question 1 (taking $K = [L]$) shows that $(G, \{1\}) \in \mathcal{F}$, i.e. that $G \in \mathcal{F}$.

Let N be a proper normal subgroup of G contained in $[L]$. We will show that the only element of N is the identity of G, i.e. $[\lambda]$. It is enough to show then that, for any $w \in \Sigma^*$ such that $[w] \in N$,

$$\forall x, y \in \Sigma^* \quad (xwy \in L \iff xy \in L).$$

Note that for any word u, $[u] \in [L]$ if and only if $u \in L$: a word in L and a word outside L can never be $\sim_L$-equivalent, as we can just take $x = y = \lambda$ in the definition of equivalence.

So now $xwy \in L \iff [x][w][y] \in [L]$. But by our choice of w, and because N is contained in $[L]$, we have:

$$\begin{aligned} xwy \in L &\iff [x][w][y] \in [L] \\ &\iff [x][y]N \in L \\ &\iff [x][y] \in L \\ &\iff xy \in L. \end{aligned}$$

and we are done.

($\Leftarrow$) : Set $L = L(G, K, \phi)$. We will show that the syntactic monoid of L is in fact G/K_G. Since $[L] = K/K_G$ is then a subgroup of G/K_G, a positive answer to Question 2 means that $G/K_G \in \mathcal{F}$, i.e. that $(G, K_G) \in \mathcal{F}$.

Let $f : G \to \Sigma^*/\sim_L$ be the monoid homomorphism defined by $g \mapsto [w]$ where $\phi(w) = g$. We will show that this is a well-defined and onto map whose kernel is K_G.

It is well-defined as, if $\phi(w) = \phi(u)$, then for any $x, y \in \Sigma^*$,

$$\begin{aligned} xwy \in L &\iff \phi(xwy) \in K \\ &\iff \phi(x)\phi(w)\phi(y) \in K \\ &\iff \phi(w) \in \phi(x)^{-1}K\phi(y)^{-1} \\ &\iff \phi(u) \in \phi(x)^{-1}K\phi(y)^{-1} \\ &\iff \phi(x)\phi(u)\phi(y) \in K \\ &\iff \phi(xuy) \in K \\ &\iff xuy \in L. \end{aligned}$$

So $[w] = [u]$. It is clearly onto.

It remains to show that $g \in K_G$ if and only if $f(g) = [\lambda]$. Assume that

$g \in K_G$, and let w be such that $\phi(w) = g$. Then for any $x, y \in \Sigma^*$,

$$\begin{aligned} xwy \in L &\iff \phi(x)\phi(w)\phi(y) \in \phi(L) = K \\ &\iff g \in \phi(x)^{-1} K \phi(y)^{-1} \end{aligned}$$

On the other hand, since $g \in K_G$, we know that $g \in \phi(x)^{-1} K \phi(x)$. Now the element $\phi(x)g$ is in two cosets $K\phi(y)^{-1}$ and $K\phi(x)$. Since cosets are either disjoint or equal, it must be that $K\phi(y)^{-1} = K\phi(x)$, from which we get $\phi(x)\phi(y) \in K$, i.e. $\phi(xy) \in K$. By definition of L, this means exactly that $xy \in L$.

Conversely, say $\phi(w) = g$ and $[w] = [\lambda]$. Then for all $x, y \in \Sigma^*$,

$$xwy \in L \iff xy \in L.$$

Choosing x and y such that $\phi(x) = \phi(y)^{-1}$ ranges over all elements of G, we get that $g = \phi(w) \in h^{-1}Kh$ for all $h \in G$. So $g \in K_G$. □

In fact, because of Proposition 65, Question 1 effectively just asks the following:

$$\text{If } (G, K) \in \mathcal{F} \text{ and } K \text{ is core-free, is } G \in \mathcal{F}?$$

The answer to our questions is clearly "yes" for regular pairs: If (G, K) is a regular pair, then the index of K in G is finite. Then $L(G, K_G, \phi)$ is just the intersection of finitely many sets of the form $L(G, g^{-1}Kg)$, all of which are also regular. So $(G, K_G) \in \mathcal{RECG}$ (note that we could have also proven this without recourse to formal language theory: the normal core of a finite index subgroup also has finite index).

In [53], Claas Röver gives an example of a deterministic context-free language whose syntactic monoid is a group which is not context-free. With some very slight assumption we can turn this into a counterexample to Question 2 for context-free languages. The construction below is entirely due to Röver.

Take a deterministic context-free language $L \subseteq \Sigma^*$ whose syntactic monoid is a group, which we call G. Since the syntactic monoid of a language is also the syntactic monoid of its complement, we can assume without loss of generality that $\lambda \in L$. Take H to be a context-free group generated by a finite symmetric generating set Ω (without loss of generality, $\Omega \cap \Sigma = \emptyset$).

Let $\pi : (\Sigma \cup \Omega)^* \to \Omega^*$ be the natural projection. Define also $A_\lambda(w)$ for any $w \in (\Sigma \cup \Omega)^*$ to be the subsequence of w consisting of all occurrences of xs in Σ such that $\pi(p_x(w)) \in W_\Omega(H)$ where $p_x(w)$ is the prefix of w up to that occurrence of x.

Now we define

$$L' := \{w \in (\Sigma \cup \Omega)^* : \pi(w) \in W_\Omega(H) \text{ and } A_\lambda(w) \in L\}.$$

Röver shows that for any deterministic context-free language L as above (i.e. whose syntactic monoid is a group G and which contains λ) and for any virtually free H as above, we have that

1. The syntactic monoid of L' constructed as above is the wreath product group $G \wr H$ of G and H, and

2. L' is a deterministic context-free language.

What exactly a wreath product is is not very relevant here, so we omit the definition (all we need to know, for later, is that the wreath product of $\mathbb{Z}$ with a finite group cannot be context-free - see [53]).

So we have:

Proposition 76. *There exists a context-free language whose syntactic monoid is a non-context-free group and whose image under the natural map* [ ] *(sending each word to its syntactic congruence class) is a subgroup of the syntactic monoid.*

Proof. Let L be the word problem of $\mathbb{Z}$. This is clearly a deterministic context-free language, and by definition it contains the empty word. Take

H to be any finite group. Follow Röver's construction described above to obtain L'. Note that $[L']$ is indeed a group:

- Clearly $\lambda \in L'$ so $[L']$ contains the identity.
- Let $u, v \in L'$. We want to show that $uv \in L'$. Since $\pi(u)$ and $\pi(v)$ are in $W_\Omega(H)$, $\pi(uv) \in W_\Omega(H)$ as well. Also, $A_\lambda(uv) = A_\lambda(u)A_\lambda(v)$. This is because there can be no occurrence of $x \in \Sigma$ in $A_\lambda(v)$ such that $\pi(up_x(v)) \in W_\Omega(H)$ but $\pi(p_x(v)) \notin W_\Omega(H)$, as $\pi(u) \in W_\Omega(H)$ by definition. Since L is closed under concatenation, $A_\lambda(uv) \in L$.
- We want to show that if $u \in L'$, then $u^{-1} \in L'$. For any occurrence of $x \in \Sigma$ in u, if the image under π of the prefix $p_x(u)$ is in the word problem, then so is the image of the suffix $s_x(u)$. Also, since $\pi(u) =_H 1$, $A_\lambda(u^{-1}) = (A_\lambda(u))^{-1}$. Since L is closed under inversion, $u^{-1} \in L'$.

L' is thus our counterexample to Question 2 in the case of context-free groups, as the wreath product $\mathbb{Z} \wr H$ is not context-free. □

As a side note, it can be shown that if both G and H are $\mathbb{Z}$, then the resulting syntactic monoid group of L' is a soluble group. A group K is *soluble* if it has a series $\{1\} = K_0 \trianglelefteq K_1 \trianglelefteq \ldots \trianglelefteq K_n = K$ such that the factors K_{i+1}/K_i are abelian for all i.

A polycyclic group is a soluble group where every subgroup is finitely generated, or equivalently where we have $\{1\} = K_0 \trianglelefteq K_1 \trianglelefteq \ldots \trianglelefteq K_n = K$ such that the factors K_{i+1}/K_i are cyclic for all i. All finitely generated nilpotent groups (see Section 6.3 for the definition of nilpotent groups) are polycyclic and hence soluble (see [52] for a proof). If G is $\mathbb{Z}$ and H is $\mathbb{Z}_2$, then the syntactic monoid of L' above is a polycyclic group which is not context-free. Thus the answer to Question 1 for polycyclic groups and context-free pairs is negative.

An interesting question would be to find out the answer to Question 1 for nilpotent groups and context-free pairs.

4.6 Ends of pairs of groups

4.6.1 General facts about ends

Recall from Chapter 2 the definition of the number of ends of a graph. It is only to be expected that if two graphs have the same "coarse geometry", then they have the same number of ends. We can finally prove Theorem 24, as we now know that any two Cayley graphs of the same group are quasi-isometric.

Theorem 77. *Any two quasi-isometric graphs have the same number of ends.*

Proof. This is a well-known result, with a straightforward but long proof. Let Γ and $\overline{\Gamma}$ be quasi-isometric, with quasi-isometry f and quasi-isometry constant α. Let their underlying vertex sets be X and Y respectively, and origins be o and $\bar{o}$ respectively. Thus $f(o) = \bar{o}$.

Let $x, x' \in X$ be in distinct connected connected components of $\Gamma^{(k)}$ for some k (recall that $\Gamma^{(k)}$ is the graph with a ball of center o and radius k taken out). We want to show that their images are in distinct connected components of $\overline{\Gamma}^{(l)}$ for some l.

Let $\bar{\pi}$ be a path in $\overline{\Gamma}$ between $f(x)$ and $f(x')$. As in Lemma 58, for any node on $\bar{\pi}$ which is not in $f(X)$ there is a node in $f(X)$ which is no further than α away from it, by the second property of quasi-isometry. So we have a diagram of the form:

$$\begin{array}{ccccccccc} f(x) & \longrightarrow & \ldots & \longrightarrow & z & \longrightarrow & z' & \longrightarrow & \ldots \longrightarrow f(x') \\ & & & & \vdots & & \vdots & & \\ & & & & \leqslant\alpha \downarrow & & \leqslant\alpha \downarrow & & \\ & & & & f(w) & \overset{\leqslant 2\alpha+1}{\dashrightarrow} & f(w') & & \end{array}$$

Let π be a shortest path in Γ connecting x and x' passing through all nodes w such that $f(w)$ is as above (i.e. either on $\bar{\pi}$ itself or at most α away from it), and furthermore such that all portions of π between such nodes w

are the shortest possible. By definition (as x and x' are in different connected components of $\Gamma^{(k)}$), there must be a node u on π such that $d_\Gamma(o,u) \leqslant k$. We know that there is a bound to how far $f(u)$ can be from $\bar{o}$, by the second half of the first part of the quasi-isometry definition:

$$d_{\overline{\Gamma}}(\bar{o}, f(u)) \leqslant \alpha d_\Gamma(o,u) + \alpha = \alpha k + \alpha.$$

Without loss of generality say that the location of u on π is between w and w' as above. Then

$$d_\Gamma(w,u) \leqslant d_\Gamma(w,w') \leqslant \alpha(d_{\overline{\Gamma}}(f(w),f(w')) + \alpha) \leqslant \alpha((2\alpha+1)+\alpha) = 3\alpha^2 + \alpha.$$

where the second inequality is a consequence of the first half of (1) in the definition of quasi-isometry. Using the second half of (1) we get

$$d_{\overline{\Gamma}}(f(w),f(u)) \leqslant \alpha d_\Gamma(w,u) + \alpha = 3\alpha^3 + \alpha^2 + \alpha.$$

Let z be the closest node to $f(w)$ on $\bar{\pi}$ - so either z and $f(w)$ are equal or at most α away from each other.We will now show that z cannot be too far away from $\bar{o}$.

$$d_{\overline{\Gamma}}(\bar{o},z) \leqslant d_{\overline{\Gamma}}(\bar{o},f(u)) + d_{\overline{\Gamma}}(f(u),f(w)) + d_{\overline{\Gamma}}(f(w),z) \leqslant 3\alpha^3 + \alpha^2 + (k+3)\alpha.$$

This means that if x, x' are in different connected components of $\Gamma^{(k)}$, then $f(x)$ and $f(x')$ are in different connected components of $\overline{\Gamma}^{(l)}$ where $l = 3\alpha^3 + \alpha^2 + (k+3)\alpha$.

We have just shown that $cc(\Gamma^{(k)}) \leqslant cc(\overline{\Gamma}^{(l)}) = cc(\overline{\Gamma}^{(3\alpha^3+\alpha^2+(k+3)\alpha)})$ for all $k \in \mathbb{N}$.

Since quasi-isometry is an equivalence relation, we also know that

$$cc(\overline{\Gamma}^{(l)}) \leqslant cc(\Gamma^{(3\beta^3+\beta^2+(k+3)\beta)})$$

for some $\beta \geqslant 1$. So

$$cc(\Gamma^{(k)}) \leqslant cc(\overline{\Gamma}^{(l)}) \leqslant cc(\Gamma^{(3\beta^3+\beta^2+(k+3)\beta)}).$$

By the sandwich lemma the sequences must have the same limits, which means that the two graphs have the same number of ends. □

Using Lemma 58, the following corollary falls out:

Corollary 78. *Any two Schreier graphs of the same pair of groups (or in particular any two Cayley graphs of the same group) have the same number of ends.*

We can thus say *the number of ends of a group* or *of a pair of groups* without reference to any generating set.

For ends of single groups/Cayley graphs, options are restricted in terms of ends. It is clear that finite groups have no ends. The free abelian group $\mathbb{Z} \times \mathbb{Z}$ on two generators has one end (its Cayley graph with respect to the natural generating set is an infinite grid), and it is easy to see that the free group on two generators has infinitely many ends. However these are the only possible numbers of ends: a finitely generated group cannot have an arbitrary number of ends. The well-known Freudenthal-Hopf theorem states that

Theorem 79 (Freudenthal-Hopf)**.** *A finitely generated group has either 0,1,2 or infinitely many ends.*

Proof. See Theorem 11.27 page 211 in [41] for a nice proof. □

Unfortunately, not much can be said of the algebraic properties of a group if it has one end. However, two-ended groups and groups with infinitely many ends are a bit special and have their own classification.

Theorem 80. *Let G be a finitely generated group. Then*

1. *G has two ends if and only if it is virtually $\mathbb{Z}$.*

2. *G has infinitely many ends if and only if either*

   - *$G = H *_F K$ where F is a proper finite subgroup of H and K with index strictly greater than 2 in both, or*
   - *$G = *_F H$ where F is a proper finite subgroup of H.*

Proof. (1) is an important result of Hopf's [27] - see Corollary 11.34 of [41] for a nice proof. The second is called Stallings' Theorem on ends of groups: see [56]. □

We thus have a nice classification of two-ended groups as one-counter groups.

Unfortunately, when we go to ends of pairs of groups, the situation gets a bit more complicated. To begin with, pairs of groups can have as many ends as they like: see example 2.3 page 187 in [54].

However, with a quite strong restriction, we do get back to the situation for Cayley graphs. The following result was proved by Houghton in [28] for locally compact groups - we cite it here for our case, where all groups are discrete. For a subgroup K of G, the *normalizer* $N_G(K)$ of K in G is the largest subgroup of G in which K is normal.

Theorem 81. *If G and K are both finitely generated, and K has infinite index in its normalizer $N_G(K)$, then the pair (G, K) has 1, 2 or infinitely many ends. Furthermore, if the pair has two ends, then we can find G' and K' such that:*

1. *$K \leqslant K' \trianglelefteq G' \leqslant G$,*

2. *$[K' : K], [G : G'] < \infty$,*

3. *G'/K' is either $\mathbb{Z}$ or the infinite dihedral group $\mathbb{Z}_2 * \mathbb{Z}_2$.*

In any case, there are various properties of ends of pairs of groups that can be stated:

Theorem 82. *Let G be finitely generated, K an arbitrary subgroup of G. Then*

1. $e(G, \{1_G\}) = e(G)$.
2. *If $N \trianglelefteq G$, then $e(G, N) = e(G/N)$.*
3. *If $[G : H] \leqslant \infty$, then $e(G, K) = e(H, K)$.*
4. *If $N \trianglelefteq G$ and $N \leqslant K \leqslant N$, then $e(G, K) = e(G/N, K/N)$.*
5. *If $[K : H] \leqslant \infty$, then $e(G, K) \leqslant e(G, H) \leqslant [K : H]e(G, K)$.*

Proof. (1) and (2) are clear from the properties of Cayley graphs. So is (3): it is a direct consequence of quasi-isometry, Lemma 58. (4) is a consequence of Proposition 52. For a proof of (5) see Lemma 1.7 in [54]. □

As a remark, an example which shows that we do have a strict inequality in item (5) rather than equality between $e(G, K)$ and $e(G, H)$ is as follows: Take G to be the dihedral group with the natural generating set (one generator of order two acting by inversion on the other which is of infinite order - apart from $\mathbb{Z}_2 * \mathbb{Z}_2$, this group can also be written as $\mathbb{Z} \rtimes \mathbb{Z}_2$), and take K to be the subgroup of order two, H the trivial subgroup. Then $e(G, K) = 1$ but $e(G, H) = e(G) = 2$.

4.6.2 Ends and language classes for pairs

We already know that a pair (G, K) is regular if and only if K has finite index in G, if and only if $\Gamma(G, K)$ has no ends.

We saw in Section 4.3 that for context-free pairs of groups, we can also somewhat characterise the space of ends:

Theorem 83. *If $(G, K) \in \mathcal{CF}$, then any Schreier graph of (G, K) has only finitely many isomorphism classes of connected components.*

This is a result from [42] that we make more explicit here. Let Γ be a finitely generated labelled graph with designated origin o. For any vertex v in the graph, one can define the connected component $\Gamma(v)$ to be the connected component of $\Gamma - B(o, |v|)$ containing v. Define also the *boundary* $\Delta(v)$ of this connected component as all vertices in $\Gamma(v)$ such that $|u| = |v|$. Now, an *end-isomorphism* between connected components $\Gamma(v)$ and $\Gamma(v)$ is a labelled graph isomorphism which sends the boundary $\Delta(v)$ to $\Delta(u)$, and a context-free graph is one which has only finitely many isomorphism classes in the set $\{\Gamma(v) : v \in V(\Gamma)\}$. As an example with infinitely many ends, look at the Cayley graph of the free group of rank two under the natural generating set. For any vertex v, the boundary of $\Gamma(v)$ is just the singleton $\{v\}$, and the connected component $\Gamma(v)$ itself is just the subtree of the Cayley graph cut off at v. All $\Gamma(v)$s thus look identical, like a ternary tree - there is only one isomorphism class of connected components.

We would like to find a similar result for ends of one-counter groups: is a pair of groups with two ends always one-counter, and does a (non-regular) one-counter pair always have two ends? The answer to the second question turns out to be no: Example 88 below gives a pair which is one-counter but has only one end. We therefore replace it by a new conjecture: a one-counter pair has at most two ends.

For the question of whether a one-counter pair has at most two ends, we would need something like the following conjecture:

Conjecture 84. *Let G be infinite, $K \leqslant G$ such that $[G : K] = \infty$. Let (G, K) be a one-counter pair. Then there is a subgroup H of finite index in G and a normal subgroup $N \trianglelefteq G$ such that*

1. *$N \subseteq K \cap H$, and*
2. *$H/N \simeq \mathbb{Z}$.*

If Conjecture 84 held then we would have:

Conjecture 85. *Let (G, K) be a one-counter pair, $[G : K] = \infty$. If Conjecture 84 is true, then there is an $N \trianglelefteq G$ such that $N \leqslant K$ and G/N is one-counter.*

Proof. We are assuming Conjecture 84, so there are H and N satisfying (1) and (2). Since $H/N \leqslant G/N$ and $[G/N : H/N] < \infty$, G/N is virtually cyclic and therefore one-counter. □

Remark 86. *Conjecture 85 and Conjecture 84 preceding it are in fact equivalent. If we have Conjecture 85, then we know that G/N, being one-counter and infinite, has an infinite cyclic subgroup of finite index. This is a subgroup of the form H/N for some subgroup $H \leqslant G$ containing N, and as $[G/N : H/N] = [G : H]$, the index of H in G must be finite. Clearly $N \subseteq K \cup H$, so we get Conjecture 84.*

Theorem 87. *Assume that Conjecture 84 holds. If (G, K) is a OC pair, then the Schreier graph $\Gamma(G, K, A)$ of G and K (with respect to any generating set A) has at most two ends.*

Proof. If $[G : K] < \infty$, then the Schreier graph of G and K is finite and therefore has no ends. So we can from now on assume that K has infinite index in G.

By Conjecture 85, we have a normal subgroup N of G contained in K such that $G' := G/N$ is one-counter. By Lemma 71, either $|K/N| < \infty$ or $[G/N : K/N] < \infty$. We can ignore the second case, because it implies that $[G : K]$ is finite. So assume we are in the first case. Set $K' = K/N$.

The situation is then the following: we have a one-counter group, G', with a finite subgroup K'. Since G' is one-counter, it is virtually cyclic; in fact it is virtually $\mathbb{Z}$, since otherwise G/N would be finite, contradicting our assumption that $[G : K] = \infty$. Let H' be a subgroup of finite index in G' isomorphic to $\mathbb{Z}$. We can assume that H' is normal in G': if it is not, look

at $M = g_1^{-1}H'g_1 \cap \ldots \cap g_n^{-1}H'g_n$ where $g_1, \ldots, g_n$ are coset representatives of H' in G. Clearly $g^{-1}Mg = M$ for all $g \in G'$, so M is normal in G', and M has finite index in G' since it is the intersection of finitely many finite-index subgroups. Since any finite subgroup of $\mathbb{Z}$ is isomorphic to $\mathbb{Z}$ itself, we still have an infinite cyclic group, and it still has finite index in G'; so replace H' by M if necessary.

Note that any nontrivial element of H' has infinite order, so no such element can be in K' since the latter is a finite subgroup. We now have two trivially intersecting subgroups, one of which is normal. So set

$$G'' := H' \rtimes K'.$$

This semidirect product has finite index in the whole group G', because H' does. Now by Lemma 77 (and the fact that the Schreier graph of $(G/N, K/N)$ and the Schreier graph of (G, K) are isomorphic), we know that

$$e(G'', K') = e(G', K') = e(G/N, K/N) = e(G, K).$$

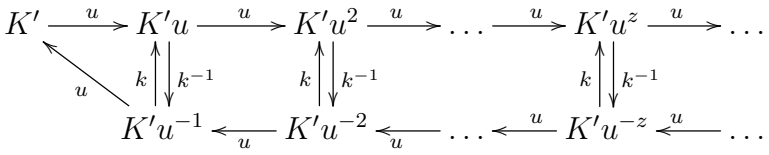
Hence it is enough to show that the Schreier graph of the pair (G'', K') has at most two ends. Let us examine this graph. As a finite generating set, choose $K' \cup \{u\}$ where u generates H'. The nodes of the graph are of the form Ku^z with $z \in \mathbb{Z}$. It is obvious how u acts on the coset space, but what can we say about the elements of K'?

In a semidirect product, there is always a group homomorphism from the non-normal subgroup to the group of automorphisms[3] of the normal subgroup, defined by conjugation (see [52] page 27). So here there is a group homomorphism $\theta : K' \to Aut(H')$ defined by $\theta(k)(h) = khk^{-1}$ for all $k \in K', h \in H'$ (we call this the action of k on h). But $\mathbb{Z}$ has only two automorphisms, so each $k \in K'$ acts on H' either as the identity map or the

[3]An *automorphism* of a group is an isomorphism from the group to itself. The set of automorphisms of a group form a group under composition of functions.

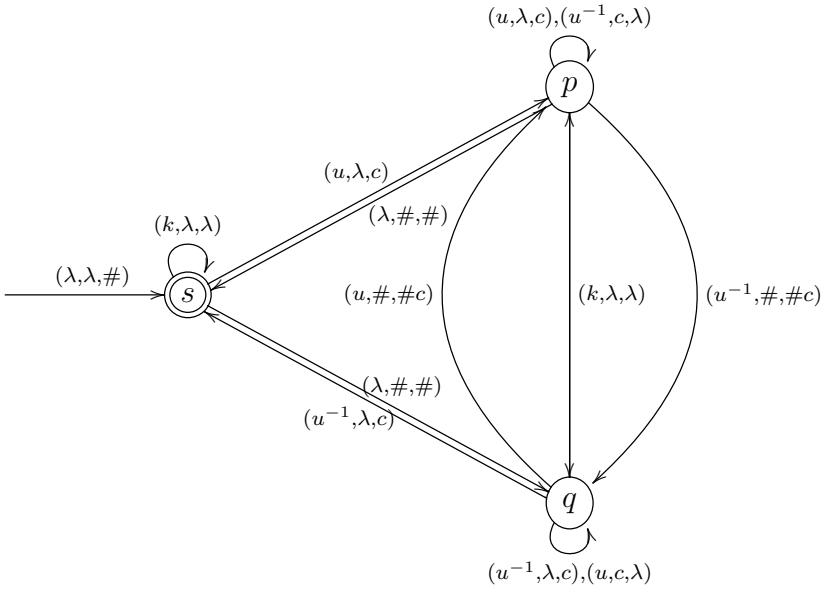
inversion map. Note that if k acts as inversion, then so must k^{-1}, as θ is a group homomorphism.

If K' has elements acting as inversion, then the Schreier graph will look as follows (ignoring reflexive loops, and where the arrows labeled k and k^{-1} exist for each element of K' acting as inversion):



and this clearly has one end. If there is no element acting as inversion on H', then the Schreier graph of (G'', K') (again ignoring reflexive loops) is isomorphic to that of $\mathbb{Z}$ and therefore has two ends. □

Example 88. *Let $G = \langle u, k | k^2 = 1, kuk = u^{-1} \rangle$ be the infinite dihedral group, and let $K = \langle k \rangle$ be the subgroup of order two generated by k. Then (G, K) is a one-counter pair, and a one-counter automaton recognizing it is shown below:*



Intuitively, if we are in state p we are in a coset of type Ku^i for some $i \geqslant 0$, and if we are in q we are in a coset of type Ku^{-i} for $i \geqslant 0$. When the stack is empty, these two states are equivalent. The action of u modifies the stack, and the action of k swaps between p and q without changing the stack, which corresponds to swapping between Ku^i and Ku^{-i}. The coset K is represented by the state s, at which point the stack must be empty.

This pair has only one end: its Schreier graph is exactly the one shown in the previous theorem, except that K' is replaced by K.

Chapter 5

Groups with a recursively enumerable irreducible word problem

5.1 Introducing irreducible word problems

In this chapter we consider the irreducible word problem of a group; this is the set of words in the word problem W which have no non-empty proper subwords in W (see Definition 94). The irreducible word problem is intrinsically connected with the word problem; using the terminology of [29] we have that W is the insertion closure of the irreducible word problem I together with the empty word $\{\lambda\}$, and I is the insertion base of W (see [13] for further details).

The notion of an irreducible word problem was introduced in [20] where groups with a finite irreducible word problem were studied. The history of the study of the irreducible word problem has, in some sense, followed that of the word problem in the reverse direction; with the word problem the initial considerations were with groups with a recursive or recursively enumerable word problem and the cases where the word problem lay in a more restricted

class of languages came later. The study of the irreducible word problem started with groups with a finite irreducible word problem in [20], which was continued in [47, 48]; it was then pointed out in [13] that there are no groups whose irreducible word problem is regular but not finite. There are also some interesting connections with string rewriting systems as explained in [39]. Groups with a context-free irreducible word problem were considered in [12, 13, 14] and with a context-sensitive irreducible word problem in [35].

In the case of recursive languages it was shown in [14] that the irreducible word problem of a group is recursive if and only if the word problem is recursive. As we point out in Remark 97, this gives an example where the irreducible word problem lying in a class $\mathcal{F}$ of languages is independent of the choice of finite generating set for the group in question. While this also holds for context-sensitive languages (see [35]) it does not hold in general even when $\mathcal{F}$ is closed under inverse homomorphism, which is a particular complication when studying irreducible word problems.

In this chapter we take the next natural step by considering groups which have a recursively enumerable irreducible word problem; this is a situation where membership could possibly depend on the choice of finite generating set (see Question 101). Our main result (Theorem 100) considers the case where we do have independence and is rather surprising: having a recursively enumerable irreducible word problem with respect to every finite generating set is equivalent to having a recursive word problem. In such groups having a recursively enumerable irreducible word problem and having a recursive irreducible word problem are equivalent, which is in complete contrast to the situation for word problems as mentioned above.

The structure of the chapter is as follows. In Section 5.2 we introduce the notion of a loopy group (Definition 89) which plays a fundamental role in proving our results. We show that any group of order at least three is loopy with respect to some finite generating set (Theorem 92) and give an alternative characterisation of loopiness (Theorem 93). In Section 5.3 we

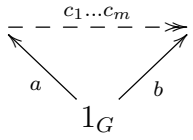
consider irreducible word problems and establish our main result. We then ask whether having a recursively enumerable irreducible word problem can depend on the choice of finite generating set in Section 5.4 and go on to investigate the potential properties of groups which do have this dependence. The material in Sections 5.2, and 5.3 has been published in [50].

5.2 Loopy groups

We now introduce the idea of a 'loopy group' which will play a central role in establishing our results. This concept appears not to be in the literature and hence the terminology is ours.

To avoid trivial situations we confine our attention to groups with at least three elements (this is not a real restriction, as when talking about groups with a recursively enumerable irreducible word problem we are really only interested in infinite groups anyway. We could potentially add $\mathbb{Z}_2$ to the definition below with no adverse effects).

Definition 89. *Let A be a finite generating set for a group G with $|G| > 2$ and let $\Sigma = A \cup A^{-1}$. G is said to be* loopy *with respect to A if, there are $a, b \in \Sigma$ such that $ab^{-1} \neq_G 1_G$, and for any such a and b there is a simple cycle in $\Gamma(G, A)$ starting at 1_G labelled by $ac_1 \dots c_m b^{-1}$ for some $c_1, \dots, c_m \in \Sigma$.*



Remark 90. *In other words, if a group is loopy, then there is a way to get from any vertex a at distance one from the origin in $\Gamma(G, A)$ to any other such vertex b without passing through the origin (if there is a path from a to b that does not pass though the origin then there is a simple such path P. Then the cycle that goes from 1_G to a, then follows P, and finishes by going*

from b to 1_G is a simple cycle of the required form). The converse of this also holds. We will use this equivalent formulation of loopiness in what follows.

Remark 91. *If G is loopy with respect to a finite generating set A then, for any edge in $\Gamma = \Gamma(G, A)$, there is a simple cycle through 1_G containing said edge. To see this consider an edge e with label $a \in \Sigma$ joining a vertex g to a vertex h. There is then an edge joining 1_G to $a = g^{-1}h$. If we pick some other element b of Σ with $ab^{-1} \neq_G 1_G$ then, by the definition of loopiness, there is a simple cycle in Γ starting at 1_G labelled by $ac_1 \dots c_m b^{-1}$ for some $c_1, \dots, c_m \in \Sigma$. The word $ac_1 \dots c_m b^{-1}$ also labels a simple cycle starting at g where the first edge in the cycle is e as required.*

A critical fact is that any finitely generated group is loopy with respect to some finite generating set:

Theorem 92. *For any finitely generated group G with $|G| > 2$ there is a finite generating set A with respect to which G is loopy.*

Proof. Let $G = \langle B|R\rangle$ be a presentation of G where B is finite. We extend the generating set B as follows: for any two elements $a, b \in B \cup B^{-1}$ such that $a^{-1}b \notin B \cup B^{-1} \cup \{1_G\}$ we define a new generator $z_{a^{-1}b}$ where $z_{a^{-1}b} =_G a^{-1}b$. Let

$$A \;=\; B \cup B^{-1} \cup \{ z_{a^{-1}b} : a, b \in B \cup B^{-1},\ a^{-1}b \notin B \cup B^{-1} \cup \{1_G\} \}$$

and

$$S = R \cup \{ z_{a^{-1}b} b^{-1} a \}.$$

It is clear that $\langle A|S\rangle$ is another presentation for G; we will show that G is loopy with respect to A. As we commented in Remark 90 all we need to show is that, for any two elements a and b in our new set A of generators such that $a^{-1}b$ does not represent the identity, there is a path from one to the other in $\Gamma = \Gamma(G, A)$ not passing through the origin. We split our consideration up

into three cases, assuming at the outset that $a^{-1}b \notin B \cup B^{-1}$, otherwise we are already done.

1. If both the generators a and b are in $B \cup B^{-1}$, then $z_{a^{-1}b}$ labels a simple path from a to b and we are done.

2. Next let us assume that one of the generators b is in $B \cup B^{-1}$ and that the other generator is in $A - (B \cup B^{-1})$; so the other generator is of the form $z_{a^{-1}c}$ for some $a, c \in B \cup B^{-1}$.

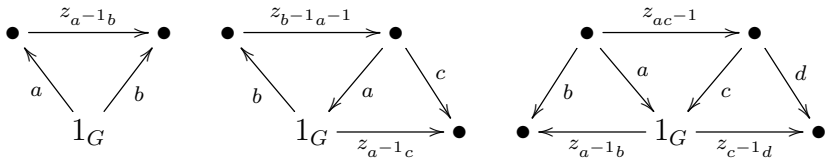
   (a) If $b \neq a^{-1}$ then $z_{b^{-1}a^{-1}c}$ labels a path from b to $z_{a^{-1}c}$ which does not pass through the origin and we are done.

   (b) If $b = a^{-1}$ then c labels a path from b to $z_{a^{-1}c}$ and we are done.

3. Finally let us assume that both the generators are in $A - (B \cup B^{-1})$; call them $z_{a^{-1}b}$ and $z_{c^{-1}d}$.

   (a) If $a \neq c$ then $b^{-1}z_{ac^{-1}}d$ labels a path from $z_{a^{-1}b}$ to $z_{c^{-1}d}$ and we are done.

   (b) If $a = c$ then $b^{-1}d$ labels a path from $z_{a^{-1}b}$ to $z_{c^{-1}d}$ and we are done.



(1) (2a) (3a)

We have covered all the possibilities and so the result is established. □

If G is a group with a finite generating set A we define a property P_A on G as follows: if $g \in G$ then $P_A(g)$ means there is a simple cycle passing through 1_G and g in $\Gamma(G, A)$. We now have the following characterisation of loopiness:

Theorem 93. *Let G be a group with a finite generating set A; then G is loopy with respect to A if and only if $P_A(g)$ holds for all $g \in G$.*

Proof. ($\Rightarrow$) : Assume that G is loopy. If $a \in \Sigma = A \cup A^{-1}$ then, as in Remark 91, there is a simple cycle containing the edge joining 1_G to a in $\Gamma = \Gamma(G, A)$ with label a; hence $P_A(a)$ holds for any generator. Given that Γ is connected, to prove that $P_A(g)$ holds for all $g \in G$ it is sufficient to prove the following:

$$\text{if } P_A(h) \text{ holds and } g \text{ is a neighbour of } h \text{ in } \Gamma$$
$$\text{then } P_A(g) \text{ holds.}$$

So assume that $P_A(h)$ holds and that g is a neighbour of h in Γ. Let L_1 be a simple cycle passing through 1_G and h in Γ. Let a be the label of the edge from h to g in Γ.

Choose a neighbour k of h on L_1 and let b be the label of the edge from h to k. Assume for a contradiction that $P_A(g)$ does not hold; we have that g does not lie on L_1 and so b is distinct from a. Since G is loopy there is a simple cycle starting at 1_G with label of the form $ac_1c_2 \dots c_n b^{-1}$. Premultiplying the vertices of the cycle by h gives a simple cycle L_2 starting at h with the same label; the first vertex (after h) on the cycle is $ha = g$ and the last vertex (before h is reached again) is $hb = k$.

Let m be the first vertex on L_2 after g that lies on L_1 (see Figure 5.1); such a vertex must exist as k lies on L_1. Let P be the subpath of $c_1 \dots c_n$ between g and m. By our choice of m, P does not intersect L_1 at any vertex except m. Therefore, to get a simple loop containing 1_G and g, all we need to do is replace the portion of L_1 between h and m by the edge from h to g followed by P. This contradicts the fact that $P_A(g)$ does not hold.

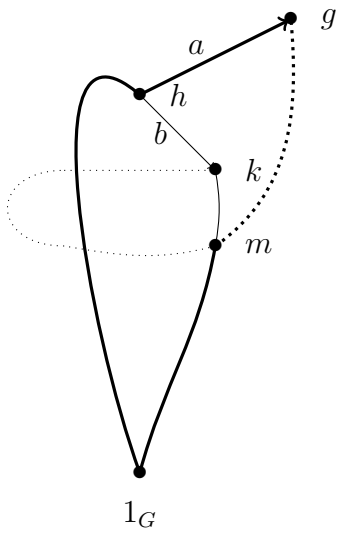


Figure 5.1: The path in bold indicates the simple loop through g; solid lines represent L_1 and bold dotted lines represent P.

($\Leftarrow$) : Let a and b be any two elements of Σ with $ab^{-1} \neq_G 1_G$. We know that $P_A(a^{-1}b)$ holds and so there are two simple paths σ and τ to the vertex $a^{-1}b$ in Γ which only have the vertices 1_G and $a^{-1}b$ in common. The vertex a^{-1} can only lie on one of these two paths and so we may assume that σ does not contain a^{-1}. Since σ is a path from 1_G to $a^{-1}b$ not containing a^{-1}, premultiplying each vertex on the path by a gives a path (with the same label as σ) from a to b not containing 1_G. As in Remark 90 we see that this ensures that G is loopy. □

5.3 Irreducible word problems

We now come to the main topic of this chapter, where we define the irreducible word problem and describe its similarities, differences and interactions with the word problem.

Definition 94. *Let A be a finite generating set for a group G. The* irreducible word problem *$I_A(G)$ of G with respect to A is the set of words $w \in W_A(G)$ such that*

$$(w = \alpha u \beta \text{ and } \alpha\beta \neq \lambda \text{ and } u \in W_A(G)) \implies u = \lambda.$$

In other words $I_A(G)$ is the set of words $w \in W_A(G)$ which have no non-empty proper subwords belonging to $W_A(G)$.

Just as the word problem for a group G with respect to a finite generating set A can be identified with the set of labels of cycles in the Cayley graph $\Gamma(G, A)$ which start and end at 1_G, the irreducible word problem (with respect to A) can be identified with the labels of the simple cycles starting and ending at 1_G, as any subloops would have subwords in the word problem as labels.

Similarly, just as we can express an arbitrary path starting and ending at 1_G in the Cayley graph as a combination of simple loops stuck to each other at certain nodes, we can express the word problem as the result of an operation on the irreducible word problem. This is the idea behind the following definitions:

Definition 95. *Let $L \subseteq \Sigma^*$ be a formal language. L is said to be* insertion closed *if for any $u, v, w \in \Sigma^*$,*

$$v \in L, uw \in L \Rightarrow uvw \in L.$$

K is said to be the insertion closure *of L if it is the smallest insertion closed language containing L.*

It is easy to see that $W_A(G)$ is the insertion closure of $I_A(G)$ for any group G and finite generating set A. Given this, it is easy to generalize an algorithm determining membership in the latter to one determining membership in the former. This is idea behind the proof of the following result from [14] (see Theorem 4.4 there):

Proposition 96. *If G is a group and A is a finite generating set for G then $I_A(G)$ is recursive if and only if $W_A(G)$ is recursive.*

Remark 97. *If $\mathcal{F}$ is a family of languages then the fact that the irreducible word problem of a group G with respect to a finite generating set A belongs to $\mathcal{F}$ depends on the choice of A even if $\mathcal{F}$ is closed under inverse homomorphisms, in contrast to the situation with the word problem. However Proposition 96 shows that this is not the case if $\mathcal{F}$ is the class of recursive languages.*

We also need the following result from [12] (see Proposition 8.3.1 there):

Proposition 98. *Let G be a group and A be a generating set for G. If $I_A(G)$ is recursively enumerable then $W_A(G)$ is recursively enumerable.*

Proof. Assume that $I = I_A(G)$ is recursively enumerable.

Let $\Sigma = A \cup A^{-1}$ and let $\mathfrak{A}$ be an algorithm that, when given an input $\alpha \in \Sigma^*$, terminates if and only if $\alpha \in I$. We outline an algorithm $\mathfrak{A}'$ which, given an input $\alpha \in \Sigma^*$, terminates if and only if $\alpha \in W := W_A(G)$.

We apply $\mathfrak{A}$ to α and every one of its non-empty proper subwords. If $\mathfrak{A}$ does not terminate on any of these words, then α is not in W and $\mathfrak{A}'$ does not terminate either. If $\mathfrak{A}$ terminates on α (in which case $\alpha \in I \subset W$), then $\mathfrak{A}'$ terminates. If $\mathfrak{A}$ terminates on some non-empty proper subword u, say $\alpha = \beta u \gamma$, then u is deleted from α. The following procedure is then repeated:

(A) The algorithm $\mathfrak{A}$ is applied to every non-empty subword of the leftover word. If $\mathfrak{A}$ terminates on some such non-empty subword u', then u' is deleted.

The algorithm $\mathfrak{A}'$ terminates with the empty word λ if and only if $\alpha \in W$. If, on some iteration of (A), $\mathfrak{A}$ does not terminate on any non-empty subword, then $\mathfrak{A}'$ does not terminate either. □

We now see that loopiness and recursive enumerability with respect to the same generating set is enough to give recursivity for the word problem. The following is important for establishing our main result:

Theorem 99. *Let G be a group generated by a finite set A such that $P_A(g)$ holds for all $g \in G$ in the Cayley graph $\Gamma(G, A)$. Then*

$$I_A(G) \text{ is recursively enumerable } \Rightarrow W_A(G) \text{ is recursive.}$$

Proof. Assume that $I = I_A(G)$ is recursively enumerable, so that we have an algorithm $\mathfrak{A}$ which terminates if and only if its input is in I.

We aim to show that both the word problem and its complement are recursively enumerable. By Proposition 98 we know that $W = W_A(G)$ is recursively enumerable and so we have an algorithm $\mathfrak{B}$ which terminates if and only if its input is in W. We now want an algorithm $\mathfrak{B}'$ which terminates if and only if its input is not in W.

Let $\Gamma = \Gamma(G, A)$ and $\Sigma = A \cup A^{-1}$. If w is any word in $\Sigma^* - W$ then, since $P_A(w)$ holds, we may choose a simple cycle containing 1_G and w in Γ. If v is the label of the simple path from 1_G to w and u the label from w back to 1_G on that cycle then we see that $wv^{-1} \in W$ and that $vu \in I$.

Our algorithm $\mathfrak{B}'$ which terminates if and only if its input w is not in W proceeds as follows:

(A) Start enumerating words which have w as a proper prefix; for each such word α we enumerate, we start $\mathfrak{B}$ on α which will terminate if α is in W. This successively generates the non-empty words v^{-1} such that $wv^{-1} \in W$.

(B) For each such word v^{-1} we generate in (A) we start enumerating words vu which have v as a proper prefix; for each such word vu we enumerate, we start $\mathfrak{A}$ which will terminate if $vu \in I$.

As above, if $w \notin W$, then such non-empty words v and u must exist and, once we have generated them and confirmed that $vu \in I$, then we know that

$v \notin W$ (by definition of I) and hence that $w \notin W$ (since $w =_G v$). So $\mathfrak{B}'$ terminates if and only if its input w does not lie in W as required. □

We now have our main result:

Theorem 100. *If G is a finitely generated group then the following are equivalent:*

1. *G has a recursively enumerable irreducible word problem with respect to every finite generating set.*
2. *G has a finite generating set with respect to which it has a recursively enumerable irreducible word problem and is loopy.*
3. *G has a recursive word problem with respect to every finite generating set.*
4. *G has a recursive irreducible word problem with respect to every finite generating set.*

Proof. $(1 \Rightarrow 2)$: Suppose G has a recursively enumerable irreducible word problem with respect to every finite generating set. By Theorem 92 there is a finite generating set A with respect to which G is loopy. Since G has a recursively enumerable irreducible word problem with respect to every finite generating set, it certainly has such an irreducible word problem with respect to A.

$(2 \Rightarrow 3)$: Suppose that G is loopy with respect to a finite generating set A and that $I_A(G)$ is recursively enumerable. By Theorem 93 we have that $P_A(g)$ holds in $\Gamma(G, A)$ for all g in G. Since $I_A(G)$ is recursively enumerable Theorem 99 gives that $W_A(G)$ is recursive. As the word problem of G being recursive is independent of the choice of finite generating set, G has a recursive word problem with respect to every generating set.

$(3 \Rightarrow 4)$: This follows immediately from Proposition 96.

$(4 \Rightarrow 1)$: This follows immediately from the fact that a recursive language is recursively enumerable. □

5.4 Dependence on generating set

In [12] it is claimed that a group can be constructed whose word problem with respect to a particular finite generating set A is recursively enumerable but not recursive. However, the proof presented there is faulty - we therefore present it as a question below (Question 101).

Question 101. *Is there a group which has a recursively enumerable but not recursive irreducible word problem with respect to one of its finite generating sets?*

Remark 102. *Suppose Question 101 had a positive answer. Then the group G in question would have two finite generating sets A and B such that $I_A(G)$ is recursively enumerable but $I_B(G)$ is not recursively enumerable. By Proposition 98 we would have that $W_A(G)$ is recursively enumerable; given that the word problem of a group being recursively enumerable is independent of the choice of finite generating set, we have that $W_B(G)$ would also be recursively enumerable. So the converse of Proposition 98 would not hold (given that $W_B(G)$ would be recursively enumerable but $I_B(T(L))$ would not).*

In what follows, we investigate what the class of groups where the irreducible word problem is recursively enumerable with respect to some but not all finite generating sets would look like if this class were non-empty. However, sentences such as the above are so cumbersome that we feel the need to introduce some sanity-preserving terminology!

Definition 103. *We say that a group G is A-*mimsy *for a finite generating set A if $I_A(G)$ is recursively enumerable. We say that G is* uniformly mimsy *if G is A-mimsy for every A, and* uniformly non-mimsy *if G is not A-mimsy for any A. We will call G* partially mimsy *if there are generating sets A and B such that G is A-mimsy but not B-mimsy.*

Recall the definitions of the Cayley graph of a group and of its number of ends (Definitions 20 and 23 respectively). Recall also that finitely generated

groups can have either 0 (if they are finite), 1, 2 (if they are virtually cyclic) or infinitely many ends. Since a group is virtually cyclic if and only if its word problem is one-counter, it transpires that every two-ended group has solvable word problem, and hence is uniformly mimsy. The same is true of course for finite groups.

Unfortunately, the situation for one-ended and infinitely ended groups is not nearly as clear-cut. On the other hand, we will now see that, with regards to being mimsy, groups with one end act rather tamely. Intuitively, this is because such groups have Cayley graphs that are highly connected. Indeed, we will show below that groups with one end are loopy with respect to every generating set. By the results in Section 5.3, this will mean that as soon as we have mimsiness with respect to a certain generating set, it will propagate to all generating sets.

We first need to define some properties of graphs:

Definition 104. *An* n-separation *of a connected undirected graph is a set of* n *nodes whose removal (together with any adjacent edges) disconnects the graph. A graph is said to be* n-separable *if there is an* n*-separation and* n *is the smallest such natural number. A graph is* n-connected *if it has no* $(n-1)$*-separation. The* connectivity *of a group is the minimum connectivity over all its Cayley graphs.*

The theorem below is from [8].

Theorem 105. *A group possesses a 1-separable Cayley graph if and only if it is either infinite cyclic or a nontrivial free product of a finite rank free group and groups of connectivity two or more.*

We remark here that all the groups of connectivity two or more are freely indecomposable (i.e. not expressible as a nontrivial free product): if a group is freely decomposable it is 1-separable.

Also note that the above theorem automatically excludes finite groups, as finite groups cannot be 1-separable. This is because every group element g

has finite order, therefore there is a simple loop passing through g. So to disconnect 1_G and g one needs to remove at least two points. This holds for any two points in the Cayley graph. This argument gives a little flavour of what is to come: loopiness sabotages 1-separability.

Lastly, nontrivial free products of groups generally have infinitely many ends, intuitively because the two free factors are unrelated, giving rise to infinitely many "branches" in the graph. The only exception to this rule is the group $\mathbb{Z}_2 * \mathbb{Z}_2$ (the infinite dihedral group) which in fact has two ends. Therefore the above theorem exclusively concerns groups which either have infinitely many ends, or are infinite cyclic or infinite dihedral.

Recall from Chapter 4 Stallings' theorem on groups with infinitely many ends, characterizing them as HNN extensions or free products with amalgamation (Theorem 80). Note that a free product is just a free product with amalgamation over the trivial subgroup. This explains the remark above about ends of free products (it is easy to see that the infinite dihedral group has two ends: its Cayley graph with respect to the presentation $\langle a, b | a^2 = 1, aba = b^{-1} \rangle$ looks like a two-way infinite ladder).

Proposition 106. *All groups with one end are loopy with respect to every generating set.*

Proof. Let G be a one-ended group. Assume G has a generating set A with respect to which it is not loopy, then by definition of loopiness, the removal of the origin in $\Gamma(G, A)$ disconnects the graph. Then by Theorem 105, G is either infinite cyclic or a nontrivial free product, all of which have either two or infinitely many ends. □

We will now show that being loopy with respect to every generating set is almost equivalent to being freely indecomposable, but first we need a simple auxiliary lemma (again folklore).

Lemma 107. *If $G = H * K$ is finitely generated, then so are H and K.*

Proof. Let $g_1, \ldots, g_n$ be a set of generators for G. Let $\pi : G \to H$ be the unique group homomorphism sending every element of H to itself and every element of K to the identity. We know such a π exists due to the universal property of free products (in other words, any element of G can be uniquely expressed as a product $h_1k_1 \ldots h_nk_n$ where $h_i \in H$ and $k_i \in K$; π sends $h_1k_1 \ldots h_nk_n$ to $h_1 \ldots h_n$). Since π is a group homomorphism, $H = \pi(G)$ is generated by $\pi(g_1), \ldots, \pi(g_n)$, so is finitely generated. The same argument of course holds for K. □

Proposition 108. *A group is freely indecomposable if and only if it is either infinite cyclic or loopy with respect to every generating set.*

Proof. ($\Leftarrow$) : Assume for a contradiction that $G = H * K$. Then G cannot be loopy with respect to the natural generating set, i.e. the union of a generating set for H and a generating set for K: there cannot be a path from a generator of H to a generator of K not passing through the identity, as this would imply the existence of a nontrivial relation between the two generators, contradicting the fact that G is a free product of H and K.

($\Rightarrow$) : If G is not infinite cyclic and not loopy with respect to a certain generating set, then it has a 1-separable Cayley graph with respect to that generating set, so it is freely decomposable by Theorem 105. □

Proposition 109. *If a group is freely indecomposable, then it is either uniformly mimsy or uniformly non-mimsy.*

Proof. If the group is infinite cyclic, then it has recursive word problem and so it is uniformly mimsy.

If on the other hand it is loopy with respect to every finite generating set, then if it is A-mimsy for some finite generating set A then it is uniformly mimsy (see Theorem 100). □

This gives us that groups with one end are either uniformly mimsy or uniformly non-mimsy. It is not difficult to think of an example of a one-ended group with solvable (i.e. recursive) word problem (which would then

be uniformly mimsy). Take $\mathbb{Z} \times \mathbb{Z}$ with presentation $\langle a, b | ab = ba \rangle$. Then the word problem is the set

$$\{w \in \{a, a^{-1}, b, b^{-1}\}^* : |w|_a = |w|_{a^{-1}}, |w|_b = |w|_{b^{-1}}\}$$

of words containing the same number of a symbols as a^{-1} symbols, and the same number of b symbols as b^{-1} symbols. This set is clearly recursive.

However, is there indeed an example of a uniformly non-mimsy group? This question is equivalent to asking whether there are any one-ended groups with unsolvable word problem. To see that there are indeed such groups, note first that the direct product of two finitely generated infinite groups has one end (see below, or [7], Theorem 6.2.3).

Theorem 110. *The direct product of two finitely generated infinite groups always has one end.*

Proof. Clearly the direct product is infinite and thus has at least one end. It is a result of Freudenthal's [15] that a group with infinitely many ends cannot be a direct product of infinite groups. It therefore remains to show that such a group cannot have two ends.

Groups with two ends are virtually $\mathbb{Z}$: let $\langle (g, h) \rangle \leqslant G \times H$ be the infinite cyclic group of finite index[1]. Then $\langle g \rangle$ and $\langle h \rangle$ both have finite index in G and H respectively: it is easy to see that

$$[H : \langle h \rangle], [G : \langle g \rangle] \leqslant [G \times H : \langle (g, h) \rangle].$$

So $\langle g \rangle$ and $\langle h \rangle$ are both infinite cyclic, because G and H are both infinite. But now, $\langle (g, 1_H) \rangle$ is isomorphic to $\mathbb{Z}$ and has infinite index in $G \times H$. This is not possible because by Lemma 71, any subgroup of a virtually cyclic group is either finite or has finite index in the virtually cyclic group. Therefore the direct product has only one end. □

[1]When x, y are elements of a set X, notation such as $\langle x, y \rangle \leqslant X$ will mean the subgroup of X generated by x and y.

Now recall that if two groups H and K have presentations $\langle A|R_H\rangle$ and $\langle B|R_K\rangle$ respectively (we can always choose A and B to be disjoint), then $G = H \times K$ has presentation $\langle A \cup B|R_H \cup R_K \cup S\rangle$ where S is a set of relators specifying that every element of A commutes with every element of B. Clearly

$$W_A(H) = W_{A\cup B}(G) \cap (A \cup A^{-1})^*.$$

Since the class of recursive languages is closed under intersection with regular languages, if the word problem of G were to be recursive then so would the word problem of H. Hence a group which is the direct product of two infinite (finitely generated) groups, one of which has unsolvable word problem, is a one-ended group with unsolvable word problem. This ends our small digression on one-ended groups.

We have seen that only groups with infinitely many ends can be partially mimsy. But can we say even more? Using the above analysis, we will deduce a theorem about the structure of partially mimsy groups. But first we will need a few well-known theorems from group theory that will make it clear why we were so interested in freely indecomposable groups.

Theorem 111 (Grushko-Neumann)**.** *If F is a finitely generated free group and $\phi : F \to *_i A_i$ is an onto group homomorphism, then F factorizes into a free product $*_i F_i$ such that $\phi(F_i) = A_i$.*

Proof. See [38], III.3, Proposition 3.7. □

Corollary 112. *Let $r(G)$ denote the rank (i.e. minimum number of generators) of G for any group G. Then*

$$r(A_1 * \ldots * A_n) = \sum_{i=1}^{n} r(A_i)$$

for any $n \in \mathbb{N}$.

That $r(A_1 * \ldots * A_n) \leqslant \sum_{i=1}^{n} r(A_i)$ is clear. For the converse, note that there is an onto group homomorphism from the free group of rank

$r(A_1 * \ldots * A_n)$ onto G, and then use the Grushko-Neumann theorem (it is clear that if ϕ is an onto group homomorphism and F_i is any group then we have $r(\phi(F_i)) \leqslant r(F_i)$).

Corollary 113 (Grushko decomposition theorem)**.** *Any finitely generated group is a free product of a finite rank free group and finitely many freely indecomposable non-free groups.*

This tells us quite a lot about the structure of partially mimsy groups, if they exist:

Theorem 114. *If G is a partially mimsy group, then it has a nontrivial finite free decomposition*

$$G = F * A_1 * \ldots * A_n$$

where F is a free group of finite rank and the A_i's are all freely indecomposable.

Moreover, at least one of the A_i's is uniformly non-mimsy.

Proof. The first part of the theorem is a direct consequence of the Grushko decomposition theorem. Take the decomposition in question. Since all of the free factors involved are either free or freely indecomposable, they are either uniformy mimsy or uniformly non-mimsy. Therefore the free product cannot be trivial: it must have at least two factors.

Moreover, if all of the A_i's were uniformly mimsy, then all of the free factors involved would have recursive word problem, and hence recursive irreducible word problem. It is easy to see that if A and B are finite generating sets for H and K respectively, then $I_{A\cup B}(H * K) = I_A(H) \sqcup I_B(K))$. So G would have a recursive irreducible word problem (namely with respect to the union of the generating sets of the factor groups), and thus it would be uniformly mimsy by Theorem 100, a contradiction. □

As a remark, note that partial mimsiness is preserved in free products with uniformly mimsy groups:

Proposition 115. *The free product of a partially mimsy group and a uniformly mimsy group is partially mimsy.*

Proof. Let H be partially mimsy, with $I_A(H)$ recursively enumerable and $I_B(H)$ not recursively enumerable for two finite generating sets A and B of H. Let K be uniformly mimsy and C a finite generating set for it.

Since $I_{A\cup C}(H * K) = I_A(H) \sqcup I_C(K)$ we have that $H * K$ cannot be uniformly non-mimsy - since $I_C(K)$ is recursively enumerable for any generating set C of K, $I_{A\cup C}(H * K)$ must also be recursively enumerable.

On the other hand, the free product cannot be uniformly mimsy, as then $W_{B\cup C}(H * K)$ would be recursive. But

$$W_B(H) = W_{B\cup C}(H * K) \cap (B \cup B^{-1})^*$$

and so $W_B(H)$ would be recursive, making $I_B(H)$ recursive as well - a contradiction.

So the free product is partially mimsy. □

Chapter 6

Groups with a Petri net word problem

We saw in Chapter 3 that Anisimov characterised the groups with regular word problem, and that Muller and Schupp showed in [43] (modulo a subsequent result of Dunwoody's [10]) that a group has a context-free word problem if and only if it is virtually free. Herbst [21] classified the groups with a one-counter word problem as being the virtually cyclic groups. This was extended in [24], where it was shown that a group has a word problem that is a finite intersection of one-counter languages if and only if it is virtually abelian. There is also an interesting result in [11], where it is shown that a group has a word problem that is accepted by a blind counter machine if and only it is virtually abelian; see Section 6.5 for the definition of such an automaton and for some further discussion.

Whilst other classes of languages have been investigated, there are very few complete characterizations. We add to this by investigating groups whose word problem is a terminal Petri net language and establishing the following:

Theorem 116. *A finitely generated group G has a word problem that is a terminal Petri net language if and only if G is virtually abelian.*

Just like Muller and Schupp's characterisation, which uses Stallings' clas-

sification of groups with more than one end, our proof uses quite heavy group theoretic paraphernalia - in particular Gromov's theorem about groups with polynomial growth. It would be nice to have a purely language-theoretic proof, but we haven't as yet been able to find one.

Whilst the above gives a correspondence between an important family of languages and a natural class of groups, there are many variations on Petri net languages which could potentially give rise to different classes of groups. Many of these modifications are so powerful that the class of languages is found to be equal to the class of recursively enumerable languages, but there are other interesting possibilities, such as the class obtained by allowing empty transitions (i.e. λ-transitions) in the Petri net.

The structure of this chapter is as follows. We recall some basic facts about Petri nets in Section 6.1. In Section 6.2 we comment on the equivalence of various definitions for Petri net languages and cite some previous results in Section 6.3. Given this background material, showing that a finitely generated virtually abelian group has a word problem that is a Petri net language is fairly straightforward, and we do this in Subsection 6.4.1. The proof of the converse is rather more involved and we provide that in Subsection 6.4.2. In Section 6.5 we comment on how this class of groups relates to certain other classes which have arisen in considering word problems. The results in Sections 6.2, 6.4 and 6.5 have been published in [51]. We then prove a standard form theorem in Section 6.6, showing that the word problem of a virtually abelian group can always be recognized by a Petri net with a single terminal marking which is the same as the initial marking.

6.1 Definitions

In this section we set out our conventions and notation for Petri nets and recall some properties of the class of languages they accept.

Definition 117. *A* labelled Petri net *is a tuple* $P = (S, T, W, m_0, \Sigma, l)$ *where:*

(i) S *is a finite set, called the set of* places*; we will assume that an order is imposed on* S *and so it will be displayed as a tuple.*

(ii) T *is a finite set disjoint from* S*, called the set of* transitions*.*

(iii) $W : (S \times T) \cup (T \times S) \to \mathbb{N}$ *is the* weight function*, assigning a multiplicity to pairs of places and transitions. If* $W(x, y) = n$ *then we will write* $x \xrightarrow{n} y$*. If* $W(x, y) = 0$ *then we say there is no arrow from* x *to* y*.*

(iv) $m_0 \in \mathbb{N}^S$*, the* initial marking*, assigns to each place a natural number.*

(v) Σ *is a finite set called the* alphabet *and the* labelling function $l : T \to \Sigma$ *assigns a label to each transition.*

Note that the function l can be extended to a funtion $T^* \to \Sigma^*$ in the natural way (setting $l(\lambda) = \lambda$)[1].

As usual we will represent a labelled Petri net as a labelled directed graph, where the places are represented by circles, transitions by rectangles (in the diagrams of this chapter we will prefer to denote transitions by their labels for simplicity), the weight function by arrows and arrow multiplicities by numbers on the arrows (with no arrow drawn if the multiplicity is zero and no label if the multiplicity is one). Markings (i.e. elements of $\mathbb{N}^S$) are represented by tokens or natural numbers drawn in each place. The labelling function will be assumed to be total throughout this chapter.

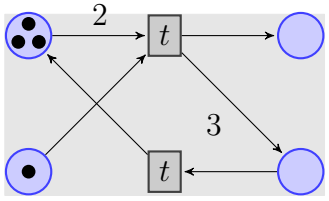


Figure 6.1: An example of a Petri net diagram.

[1]Note that l does not have to be bijective; in fact, there is a special name for the labelled Petri nets in which this holds: free Petri nets.

Now we shall describe the execution semantics of Petri nets.

Definition 118. *Let $m \in \mathbb{N}^S$ be a marking and $t \in T$ be a transition. We say that t is* enabled *at m if, for all places $s \in S$, we have $W(s,t) \leqslant m(s)$; we denote this by $m[t\rangle$. If t is enabled at m, we can fire t to get a new marking $m' \in \mathbb{N}^S$, defined by*

$$m'(s) = m(s) + W(t,s) - W(s,t)$$

for all $s \in S$. We denote this by $m[t\rangle m'$ (note that writing this automatically implies that $m[t\rangle$).

We generalize this to sequences of transitions (i.e. elements of T^*) by taking the reflexive and transitive closure in the obvious way.

We need the notion of a labelled Petri net accepting a language; there are various possibilities and we consider the "terminal language" of a Petri net. To do this we extend the definition of a labelled Petri net to include a finite set of *terminal markings* $M \subset \mathbb{N}^S$ and write

$$P = (S, T, W, m_0, M, \Sigma, l).$$

Definition 119. *The (*terminal*)* language $\mathfrak{L}(P)$ *recognized by P is the set*

$$\{l(w) : m_0[w\rangle m \text{ some } m \in M, w \in T^*\}.$$

We say that a language $L \subseteq \Sigma^*$ is a Petri net language *(PNL for short) if there is a labelled Petri net whose terminal language is L and let $\mathcal{PNL}$ denote the class of all Petri net languages.*

The class $\mathcal{PNL}$ has several nice closure properties (see references such as [31]), some of which we note here for future use:

Proposition 120. *(i) $\mathcal{PNL}$ contains all regular languages.*

(ii) $\mathcal{PNL}$ *is closed under union.*

(iii) $\mathcal{PNL}$ *is closed under intersection.*

(iv) $\mathcal{PNL}$ *is closed under inverse GSM mappings (and, in particular, under inverse homomorphisms).*

(v) $\mathcal{PNL}$ *is closed under* λ*-free GSM mappings (and, in particular, under* λ*-free homomorphisms).*

In the terminology of Chapter 3, $\mathcal{PNL}$ is therefore a semi-AFL, though it is not full as it is not closed under arbitrary homomorphisms.

The last four properties of Proposition 120 are defined in Chapter 3. For the first, note that any NFA can be transformed into a Petri net: we let the states be places, the labels of transitions in the NFA be transitions in our Petri net, the initial marking be a token in the place corresponding to the start state, and the final markings the equivalent for places corresponding to the accept states of the NFA.

Note that some authors define Petri net languages (or indeed CSSs, a very close concept), in a slightly different way: they only allow one final marking, or disallow a final marking equal to the initial one. Several clever constructions (see pages 8-21 of [19]) show that these definitions are equivalent, up to the inclusion of the empty word λ in the language. We give the details of the constructions in the next section.

6.2 Equivalence of definitions

In this section we will explicitly give the various definitions of Petri net languages given by the various authors used in our references, and we will then show that the definitions are equivalent (to each other and to ours). Many authors use the definitions interchangeably, so their equivalence is well-known but not often explicitly proven: we give the details here.

The definition in the previous section is the same as given in Jantzen's 1987 paper [31] (though he does use a different one in [30]). We will keep our terminology of labelled Petri net and PNL as above.

Below are the definitions Petersen uses to define CSS or computation sequence sets [49]:

Definition 121. *A* P-Petri net N *is a 5-tuple* (P, T, Σ, S, F) *where* P *is a finite set of places,* T *is a finite set of transitions disjoint from* P*,* Σ *is the input alphabet (or the set of labels),* $S \in P$ *is a designated start place,* $F \subseteq P$ *is a designated set of final places, and each transition* $t \in T$ *is a triple consisting of a label in* Σ*, a multiset (or bag)* I *of input places, and a bag* O *of output places.*

This is therefore almost the same as our original definition except for the designated start and final places. The labeling implies that there can be more than one transition with the same label, and the multiplicity of a place in a bag is just the multiplicity of its arrow to or from its transition in our original definition. Enabled transitions and so on are defined in the same way.

Definition 122. *The* Computation Sequence Set *of a P-Petri net is the set of all sequences of labels of transitions leading from the start marking (one token in the start place, none anywhere else) to one of the final markings (one token in one of the final places, none in any other place).*

The class of all languages which are computation sequence sets of a P-Petri net we will call CSS.

Lastly, we give Hack's definition from [19].

Hack's definition of a labelled Petri net is the same as ours (he actually splits the weight function into two separate forwards and backwards incidence functions, but this is not an essential difference).

Definition 123. *The set of* H-terminal label sequences *of a labelled Petri net N for a final marking $m_f \neq m_0$ is the set labels of sequences of transitions leading from m_0 to m_f.*

The difference between our definition and Hack's is twofold: he only allows one final marking, and this final marking cannot be equal to the start marking. His justification is that he then avoids having any H-terminal languages containing λ, as if one keeps the unique final marking condition but allows these languages to contain λ, the class of H-terminal languages of labelled Petri nets is no longer closed under union (see page 8 in [19]). Hack calls this class $\mathfrak{L}_0$ and we shall adopt this terminology.

It is known that $\mathfrak{L}_0$ and CSS are the same up to inclusion of the empty word:

Theorem 124. *For any language L, $L \in CSS \iff L - \{\lambda\} \in \mathfrak{L}_0$.*

Proof. In other words, any language in $\mathfrak{L}_0$ is in CSS (since none contain the empty word), and any language in CSS can be seen as a language in $\mathfrak{L}_0$ with the empty word added in. For a proof, see pages 19-20 of [19]. □

Now for the essential part: showing that $\mathcal{PNL}$ is the same as CSS. Clearly $CSS \subseteq \mathcal{PNL}$; the reverse inclusion is the interesting one. To show it, we will use a "standardisation" of Petri nets described in [19]. Assume we have a Petri net N; we will show it can be transformed into a P-Petri net without changing the terminal language.

1. *The run place:* Add a place named *run* with a loop to all transitions in the original net N. This clearly does not change the language. The practicality of this place is that it enables one to activate and deactivate N at will; no transitions in N can fire unless there is a token in *run*.

2. *The first transitions*: (See page 12, section 2.2.2 in [19].) There were finitely many transitions $t_1, \ldots, t_n$ enabled at the initial marking m_0 of

our labelled Petri net N. For each of these t_i we add a new transition t'_i with the same label as t_i. This t'_i will do two things: deposit the marking corresponding to $m_0[t_i\rangle$ (thus imitating t_i) and deposit a token in run, thus activating N.

3. *The start place:* Now add a place named *start* with an arrow to each of the t'_i. It is easy to see the language is not changed. We now have a designated start place and can take our new initial marking to be one token in the start place and none anywhere else.

4. *The stop transitions:* These use the same principle as the first transitions - for any final marking of our original Petri net N, there are finitely many last transitions t_i leading to them - in other words, there are finitely many t_i such that there is a marking m_j where $m_j[t_i\rangle m$ for $m \in M$ a final marking. Again, add a new transition t''_{ij} for each of these. Each t''_{ij} will both take away the token in the run place and empty the penultimate marking it is associated to (so if m_j had k tokens in place p, there will be an arrow labelled k from p to t''_{ij}). Note that this only works if the last transition is not the first transition (i.e. m_j is not m_0). In that special case, t''_{ij} has just a simple arrow from the *start* place, bypassing run altogether.

5. *The final places:* For each t''_{ij}, add a new place p_{ij}, with a simple arrow from t''_{ij} to p_{ij}. If the empty word is not in the language, taking these as the final places and assuming each final marking to be a token in a single final place and none anywhere else is enough. If λ is in the language recognized by N, simply designate the start place to be a final place as well.

The modifications above can be straightforwardly seen not to change the terminal language of the net, and allow us to transform a labelled Petri net (according to our definition) into one of Petersen's P-Petri nets.

We therefore have:

Theorem 125. $\mathcal{PNL} = CSS$,

as required.

6.3 Previous results

As we saw in Proposition 120, the class $\mathcal{PNL}$ is closed under inverse homomorphisms and intersection with regular languages (the latter fact following from parts (i) and (iii)); as a result, we have the following immediate consequence of Theorem 44:

Proposition 126. *The class of finitely generated groups with word problem a PNL is closed under taking finitely generated subgroups.*

We also have:

Proposition 127. *If G and H are finitely generated groups with word problems in $\mathcal{PNL}$ then the word problem of the direct product $G \times H$ is also in $\mathcal{PNL}$.*

Proof. If P_1 and P_2 are Petri nets recognising the word problems of G and H with respect to finite generating sets A and B respectively (where A and B are disjoint), then the disjoint union of P_1 and P_2 recognizes the word problem of $G \times H$. □

Of fundamental importance in what follows will be the so-called *Heisenberg group*, which is the group of matrices

$$\left\{ \begin{pmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} : a, b, c \in \mathbb{Z} \right\}$$

under multiplication. This is an example of a *nilpotent group*. One way of defining this concept is to let $Z(G)$ denote the *centre* of a group G

$$Z(G) := \{g \in G : gh = hg \quad \forall h \in G\},$$

and then define a series of normal subgroups $Z_1(G) \leqslant Z_2(G) \leqslant \ \ldots$ of G by:

$$Z_1(G) := Z(G), \quad Z_{i+1}(G)/Z_i(G) := Z(G/Z_i(G)) \text{ for } i \geqslant 1.$$

We say that G is *nilpotent* if $Z_i(G) = G$ for some $i \in \mathbb{N}$.

Recall the definition of 'virtually' (Definition 14). The following fact (see [25] for example) will be important here:

Proposition 128. *A finitely generated torsion-free virtually nilpotent group that does not contain the Heisenberg group is virtually abelian.*

As $\mathcal{PNL}$ is closed under union with regular languages and inverse GSM mappings, we have the following consequence of Theorem 45:

Proposition 129. *The class of finitely generated groups whose word problem lies in $\mathcal{PNL}$ is closed under taking finite extensions.*

Returning to generating sets, we say that a group G with finite generating set A has *polynomial growth* if there is a polynomial $p(x)$ such that the number of elements of G represented by words in $(A \cup A^{-1})^*$ of length at most n is bounded above by $p(n)$.

6.4 Virtually abelian if and only if PNL word problem

6.4.1 Virtually abelian implies PNL

In this subsection we prove one direction of Theorem 116, showing that a finitely generated virtually abelian group G has a word problem in $\mathcal{PNL}$. We start with the case where G is abelian:

Proposition 130. *The word problem of a finitely generated abelian group is always a PNL.*

Proof. Let G be a finitely generated abelian group. According to the structure theorem for finitely generated abelian groups, G is expressible as a direct product

$$\mathbb{Z}^r \times \mathbb{Z}/a_1\mathbb{Z} \ldots \times \mathbb{Z}/a_m\mathbb{Z}$$

where $r \geqslant 0$, $m \geqslant 0$ and $a_i = p_i^{n_i}$ for some prime p_i and some natural number $n_i \geqslant 1$. The word problem of a finite cyclic group $\mathbb{Z}/a\mathbb{Z}$ is regular (as in Theorem 46), and hence a PNL. The word problem of $\mathbb{Z}$ is a PNL as shown in the diagram below.

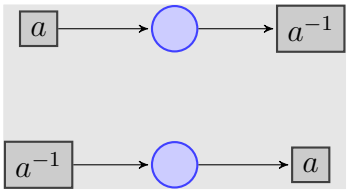


Figure 6.2: A labelled Petri net recognizing the word problem of $\mathbb{Z} = \langle a| \quad \rangle$. The empty marking is both initial and terminal, and there are no other terminal markings.

The result follows from Proposition 127. □

Propositions 129 and 130 immediately give:

Corollary 131. *Any finitely generated virtually abelian group has its word problem in $\mathcal{PNL}$.*

6.4.2 PNL implies virtually abelian

Now we consider the converse to Corollary 131 which (together with Corollary 131) will establish Theorem 116. First we prove the following:

Proposition 132. *A finitely generated group with PNL word problem has polynomial growth.*

Proof. Let G be a group generated by a finite set A and let $\Sigma = A \cup A^{-1}$. Assume that the word problem of G is recognized by $P = (S, T, W, m_0, M, \Sigma, l)$ with initial marking m_0 and set of terminal markings M.

Let us call markings which are reachable from m_0 in P and which allow the possibility of reaching a terminal marking *acceptable markings*. Notice that, given an acceptable marking m, any two sequences of transitions reaching m from m_0 must represent the same group element. This is because, if $m_0[t_1 \dots t_n\rangle m$ and $m_0[t'_1 \dots t'_k\rangle m$ and m is acceptable, then there is a sequence of transitions w from m to some terminal marking m'. But then $m_0[t_1 \dots t_n\rangle m[w\rangle m'$ and $m_0[t'_1 \dots t'_k\rangle m[w\rangle m'$ and hence both sequences of transitions are in the word problem $W_A(G)$. So

$$l(t_1 \dots t_n w) =_G 1_G =_G l(t'_1 \dots t'_k w),$$

from which we get that $l(t_1 \dots t_n) =_G l(t'_1 \dots t'_k)$.

So we have a natural mapping θ from the set of acceptable markings to G. As P recognizes the word problem of G, for each group element g there must be an acceptable marking m with $m\theta = g$, otherwise no word ww^{-1}, where w represents g, can be accepted by P. So the mapping θ is surjective.

In order to show polynomial growth, we want to show that there is a polynomial $p(n)$ such that the number of group elements represented by a sequence of generators of length n is at most $p(n)$. Since the mapping θ is surjective it is therefore sufficient to bound the number of acceptable markings reachable by a sequence of transitions of length n by such a polynomial $p(n)$.

If a sequence $t_1 \dots t_n$ reaches an acceptable marking and so does the sequence $t_{\sigma(1)} \dots t_{\sigma(n)}$ for some permutation σ of $\{1, 2, \dots, n\}$, then the two sequences reach the same marking[2]; this follows directly from the effect on a marking of firing a transition. In counting the number of acceptable mark-

[2]Recall here that the t_i are actual transitions, not labels of transitions (i.e. generators); therefore this argument does not imply that G is abelian as, for example, being able to swap labels a and b in one such sequence does not mean that we would necessarily be able to do so in all such sequences.

ings, one can therefore ignore the order in which the transitions fire: the only important thing is their multiplicity. If there are k possible transitions $t_1, \ldots, t_k$ in P, this means that there are at most as many acceptable markings induced by sequences of n transitions as there are possible values for $\mu(t_1), \ldots, \mu(t_k) \in \mathbb{N}$ such that

$$\mu(t_1) + \ldots + \mu(t_k) = n.$$

The function $\mu(t_i)$ denotes the multiplicity of the transition t_i in the transition sequence. It is now clear that the number of acceptable markings is bounded above by the polynomial $(n+1)^k$, as there are at most $n+1$ choices for each of the $\mu(t_i)$. □

Using Gromov's wonderful theorem [18] about groups with polynomial growth we immediately deduce the following:

Corollary 133. *A finitely generated group whose word problem is a PNL is virtually nilpotent.*

We now want to show that a finitely generated group whose word problem is a PNL is virtually abelian. As we will show later, it is enough to show that the Heisenberg group's word problem is not a PNL. To do this, we use the idea that a Petri net cannot "do multiplication" (in the sense described in [6]).

There are other ways to prove that the Heisenberg group does not have a PNL word problem. The result is not essentially new, and there are many alternative ways to prove it. One way would be to use Lambert's Pumping Lemma. The proof is outlined in Remark 134 below.

Remark 134. *A consequence of Lambert's Pumping Lemma (see [36]) is that for any Petri net P with initial marking m_0, final marking m_f, and letter a in the alphabet of P, the set $\mathfrak{L}(a) := \{|l(u)|_a : m_0[u\rangle m_f\}$ is infinite if and only it contains an arithmetic sequence with a non-zero common*

difference. If the language L of Theorem 137 were in $\mathcal{PNL}$, then its intersection $\{a^n b^n A^n B^n C^{n^2} : n \in \mathbb{N}\}$ with $\{a^n b^n A^n B^n C^k : n, k \in \mathbb{N}\}$ would also be in $\mathcal{PNL}$. But then $\mathfrak{L}(C) = \{n^2 : n \in \mathbb{N}\}$ would have an arithmetical subsequence with nonzero common difference, a contradiction. The rest of the proof is contained in Theorem 138.

We give below a longer and more detailed proof as we believe it provides more of the intuition. Hidden in the proof above is the use of the decidability of the reachability problem for Petri nets, which is made explicit below (see Theorem 136).

Definition 135. *A function $f : \mathbb{N}^n \to \mathbb{N}$ is said to be* computable *by a Petri net if there is a (unmarked) Petri net $P = (S, T, W)$ with places*

$$\{p_1, \ldots p_n, start, stop, p_f, run\} \subseteq S$$

and transitions go and $finish$ in T, each with multiplicity one, such that:

(i) $W(t, start) = 0$ *for all* $t \in T$.

(ii) $W(start, go) = W(go, run) = 1$.

(iii) $W(stop, t) = W(p_f, t) = 0$ *for all* $t \in T$.

(iv) $W(run, finish) = W(finish, stop) = 1$.

(v) $W(t, run) = W(run, t) = 1$ *for all* $t \in T - \{go, finish\}$.

(vi) For any initial marking m_0 such that

- *(a)* $m_0(p_i) = a_i$ *for all* $1 \leqslant i \leqslant n$,
- *(b)* $m_0(start) = 1$ *and*
- *(c)* $m_0(p) = 0$ *for all other places p (i.e. $p \in P - \{p_1, \ldots, p_n, start\}$),*

any marking m reachable from m_0 satisfies

$$\left.\begin{array}{l} m(stop) = 1, \\ m(p) = 0 \quad \forall p \in P - \{stop, p_f\} \end{array}\right\} \Rightarrow m(p_f) = f(a_1, \ldots, a_n).$$

In [6] Chrząstowski-Wachtel uses the undecidability of Hilbert's 10th problem to prove the following important result:

Theorem 136. *If multiplication is computable in a class of Petri nets then the reachability problem for that class of Petri nets is undecidable.*

By a class of Petri nets, we mean an extension of our definition of Petri nets, for example nets with inhibitor arcs or priorities (see [6]). By multiplication being computable, we mean in the sense of Definition 135 above, as a function from $\mathbb{N}^2$ to $\mathbb{N}$.

We now use Theorem 136 to prove that the word problem of the Heisenberg group cannot be a PNL. This will follow from the following result:

Theorem 137. *Let $\Sigma = \{a, b, A, B, C\}$. Then*

$$L = \{a^i b^j A^i B^j C^{ij} : i, j \in \mathbb{N}\}$$

is not a Petri net language.

Proof. We assume that L is a PNL and let $P = (S, T, W, m_0, M)$ be a Petri net recognizing L, where $S = (p_1, \ldots p_n)$, $m_0 = (a_1, \ldots a_n)$ is the initial marking and

$$M = \{(b_{11}, \ldots, b_{1n}), \ldots, (b_{m1}, \ldots, b_{mn})\}$$

is the set of terminal markings, where $m \in \mathbb{N}$, $n = |S|$ and $a_k, b_{kl} \in \mathbb{N}$ for all k and l. We now modify P to get a new Petri net P' as follows:

1. Add new places *start*, *run*, *almost stop*, q_1, q_2 and q_3.

2. Add new transitions go, $finish_1$, ..., $finish_m$ and $finish$ (each with multiplicity one).

3. Add the arrow $start \to go$.

4. For each place $p_l \in S$ (for $1 \leqslant l \leqslant n$), add the arrows $go \xrightarrow{a_l} p_l$. This means that the transition go effectively puts in place the initial marking of P.

5. For each transition $t \in T$, add the arrows $t \to run$ and $run \to t$. This means that the place run can effectively enable and disable all the firings of P, according to whether or not there is a token in it.

6. Add an arrow $go \to run$, so that go is necessary to enable the functioning of P.

7. For each of the transitions in P labelled a, add an arrow $q_1 \to a$, for each of the transitions in P labelled b, add an arrow $q_2 \to b$ and, for each of the transitions in P labelled C, add an arrow $C \to q_3$. Thus, q_1, q_2 and q_3 will effectively "count" occurrences of a, b and C respectively (though q_1 and q_2 will accumulate tokens whereas q_3 will 'eat' them).

8. For each $p_l \in S$, add an arrow $p_l \xrightarrow{b_{kl}} finish_k$. Essentially, there is a $finish_k$-transition for each of the terminal markings in M, and each of these transitions empties the places of P.

9. Finally, in order to deactivate P at the end, add arrows $run \to finish_k$, $finish_k \to almost \to finish \to stop$ for all k.

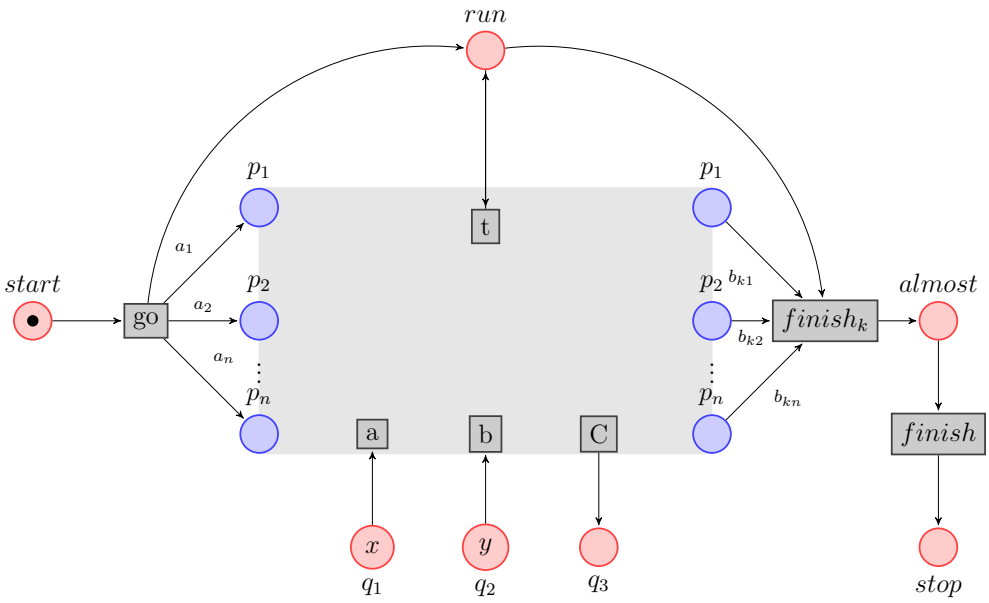


Figure 6.3: The Petri net P' at its initial marking. The grey area corresponds to the original net P. The places p_l of P are represented twice for convenience.

We let $S' = (start, run, p_1, \ldots, p_n, q_1, q_2, q_3, almost, stop)$ and define the new initial marking for P' to be

$$m'_0 = (1, 0, 0 \ldots, 0, x, y, 0, 0, 0)$$

for $x, y \in \mathbb{N}$. Let m be any (reachable) marking where every place of S' except for q_3 and $stop$ is empty. We want to show that q_3 has to hold xy tokens. Let w be the sequence of (labels of) transitions leading from m'_0 to m.

1. w must start with go, as it is the only enabled transition initially (the initial emptiness of run blocks all the other transitions).

2. The penultimate transition in w must be $finish_k$ for some k in order to empty run (recall that all places except $stop$ and q_3 must be empty at marking m). After this it is clear only *finish* can fire.

3. In between go and $finish_k$ there must be a word (recognized by P) in L: note that the only way that a $finish_k$ transition empties all places of P is if we are in the terminal marking $(b_{k1}, \ldots, b_{kn}) \in M$. Since go puts m_0 in place, we must have that $w = run\ v\ finish_k finish$ for some $k \leqslant m$ and $v \in L$.

4. In order to empty q_1 and q_2, v must have caused a and b to fire x and y times respectively. Since $v \in L$, this forces $v = a^x b^y A^x B^y C^{xy}$. This means that q_3 has xy tokens in m.

Thus, if L were recognizable by a Petri net, then multiplication would be computable. By Theorem 136, this would mean that reachability in Petri nets is undecidable. We know that this is false [40] and so L is not a PNL. $\square$

We can now immediately deduce the required result about the Heisenberg group:

Theorem 138. *The word problem of the Heisenberg group H is not a PNL.*

Proof. If a, b and c respectively denote the matrices

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

then a, b and c generate H and every relation in H can be deduced from the relations

$$ac = ca,\ bc = cb \text{ and } ab = bac$$

(this is a well known presentation for H, see [32] for example). Let W denote the word problem of H with respect to $\{a, b, c\}$. To ease clutter, we let A represent a^{-1}, B represent b^{-1} and C represent c^{-1}. We claim that the language L from Theorem 137 is just $W \cap a^* b^* A^* B^* C^*$. To see this, we recall

that $ab =_G bac$. So we get

$$\begin{array}{llllll} a^ib^j & =_G & a^{i-1}abb^{j-1} & =_G & a^{i-1}bacb^{j-1} & =_G & a^{i-1}bab^{j-1}c \\ & =_G & a^{i-1}babb^{j-2}c & =_G & a^{i-1}b^2ab^{j-2}c^2 & =_G & \cdots \\ & =_G & a^{i-1}b^jac^j & =_G & \cdots & =_G & b^ja^ic^{ij}. \end{array}$$

Now:

$$\begin{array}{rcl} a^ib^jA^kB^lC^m =_G 1 & \iff & b^ja^ic^{ij}A^kB^lC^m =_G 1 \\ & \iff & b^ja^iA^kB^lc^{ij}C^m =_G 1. \end{array}$$

It is clear that, if $i = k$, $j = l$ and $ij = m$, then $b^ja^iA^kB^lc^{ij}C^m =_G 1$. On the other hand, if $b^ja^iA^kB^lc^{ij}C^m =_G 1$, then this still holds true in the factor group $\overline{G}/\langle c\rangle$ which is a free abelian group (i.e. with no other relation other than the commuting of generators) generated by the two elements $[a]$ and $[b]$. So $[b]^j[a]^i[A]^k[B]^l =_{\overline{G}} [1]$, which gives that $i = k$ and $j = l$. So $b^ja^iA^kB^lc^{ij}C^m =_G c^{ij}C^m$, and so we must have $ij = m$ as well. So $a^ib^jA^kB^lc^{ij}C^m =_G 1$ if and only if $i = k$, $j = l$ and $ij = m$, which is what we wanted to establish.

Since the class $\mathcal{PNL}$ is closed under intersection with regular languages, we have that $W \notin \mathcal{PNL}$. □

Our result now follows:

Proposition 139. *If a finitely generated group G has a PNL word problem, then G is virtually abelian.*

Proof. We know already that G is virtually nilpotent by Corollary 133. Assume that G is not virtually abelian; then G has a nilpotent but not virtually abelian subgroup K of finite index in G. In turn, it is known (see 5.4.15 (i) of [52] for example) that K must have a torsion-free subgroup L of finite index, and L must then be nilpotent but not virtually abelian. By Proposition 128 we have that $H \leqslant L \leqslant K \leqslant G$ where H is the Heisenberg group. So H is a finitely generated subgroup of G. Since G has a PNL word problem, so does H by Proposition 126, contradicting Theorem 138. □

Taken together with Corollary 131, this completes the proof of Theorem 116.

6.5 Relationship with other language classes

In this section, we put our results into some context, comparing the class of groups with word problem in $\mathcal{PNL}$ with those in some other classes of languages, such as the class $\mathcal{CF}$ of context-free languages and the class $\mathcal{OC}$ of one-counter languages. Let $co\mathcal{CF}$ be the class of co-context-free languages (i.e., languages that are complements of context-free languages).

Recall that the one-counter languages can be thought of as those recognized by an automaton which has a counter where we store a natural number and where we can test for zero. If we have two such counters, it is well known that we can simulate a Turing machine, and so such machines accept all the recursively enumerable languages; however there are natural variations which restrict the class of languages that can be accepted. If we allow the counters to contain integers and accept a word if we can reach a designated accept state with all the counters equal to zero, then we have a BMM (*blind multicounter machine*). We could strengthen the model to allow the counters to contain natural numbers and, whilst we still cannot test if the counters are empty, transitions that attempt to decrease a counter which currently has value zero are not enabled; we again accept if we can reach a designated accept state with all the counters equal to zero. Such a machine is called a PBMM (*partially blind multicounter machine*). It was shown in [17] that every Petri net language is accepted by such a machine but that, if L is the language $\{a^n b^n : n \geqslant 0\}$, which is in both $\mathcal{OC}$ and $\mathcal{PNL}$, then L^* is not accepted by a PBMM, and hence is not in $\mathcal{PNL}$. Since $L^* \in \mathcal{OC}$, we have that $\mathcal{OC} \not\subseteq \mathcal{PNL}$. On the other hand, it is well known that $\mathcal{PNL} \not\subseteq \mathcal{CF}$ (let alone $\mathcal{OC}$); for example, the language $\{a^n b^n c^n : n \geqslant 0\}$ is in $\mathcal{PNL}$.

As we have just seen, these families of languages are both incomparable

with $\mathcal{PNL}$. However, when we turn to word problems of groups, the situation changes:

Proposition 140. *If $G \in \mathcal{OC}$ then $G \in \mathcal{PNL}$.*

Proof. A group with one-counter word problem is virtually cyclic by [21] and hence virtually abelian; the result follows from Corollary 131. □

Of course, since $\mathcal{OC} \not\subseteq \mathcal{PNL}$, finite intersections of one-counter languages are not necessarily in $\mathcal{PNL}$; however this situation also changes when we restrict ourselves to word problems. As we mentioned in the introduction to this chapter, it was shown [24] that a group has a word problem that is the intersection of finitely many one-counter languages if and only if it is virtually abelian; so we have the following:

Corollary 141. *$G \in \mathcal{PNL}$ if and only if $G \in \bigcap_{fin} \mathcal{OC}$.*

We should mention that not only is $\bigcap_{\text{fin}} \mathcal{OC}$ not a subset of $\mathcal{PNL}$, but $\mathcal{PNL}$ is not a subset of $\bigcap_{\text{fin}} \mathcal{OC}$ either. In order to prove this, we must introduce semilinear sets and the Parikh mapping.

Definition 142. *A subset X of $\mathbb{N}^n$ is said to be* linear *if there are n-tuples $\vec{v_0}, \vec{v_1}, \ldots, \vec{v_m} \in \mathbb{N}^n$ such that*

$$X = \left\{ \vec{v_0} + \sum_{i=1}^{m} m_i \vec{v_i} : m_i \in \mathbb{N} \right\}.$$

A set $Y \subseteq \mathbb{N}$ is said to be semilinear *if it is a union of finitely many linear sets.*

Let Σ be an alphabet with n symbols (we can assume they are ordered in some way $a_1, \ldots, a_n$). Then the Parikh mapping is the map which just counts the number of occurrences of each letter: $\Phi : \Sigma^* \to \mathbb{N}^n$ is defined by $w \mapsto (|w|_{a_i})_i$. Parikh's theorem essentially says that if one ignores the ordering of the letters in the words, a context-free language is just like a regular language; the precise statement is as follows:

Theorem 143 (Parikh)**.** *For any context-free language L, $\Phi(L)$ is a semilinear set.*

Proof. See Theorem 2 in [46]. □

This enables us to prove:

Proposition 144. *$L = \{a^n b^m : 1 \leqslant m \leqslant 2^n, 1 \leqslant n\}$ is in $\mathcal{PNL}$ but not in $\bigcap_{fin} \mathcal{OC}$.*

Proof. It is known that $L \in \mathcal{PNL}$; see [30]. Assume that $L \in \bigcap_{\text{fin}} \mathcal{OC}$, say

$$L = L_1 \cap \cdots \cap L_n$$

where the L_i are one-counter languages. Let K be the regular language a^*b^*.

Since $L \subseteq K$, we have $L = (L_1 \cap K) \cap \cdots \cap (L_n \cap K)$ and so, without loss of generality, we can assume that $L_i \subseteq K$ for all i (as the intersection of a one-counter language and a regular language is one-counter). Since the Parikh mapping $\Phi : \Sigma^* \to \mathbb{N}^2$ defined by $w \mapsto (|w|_a, |w|_b)$ is bijective on K, we have

$$\Phi(L) = \Phi(L_1) \cap \cdots \cap \Phi(L_n).$$

By Theorem 143, we know that any context-free language, and hence any one-counter language, has a semilinear Parikh image. Since the L_i are all one-counter, $\Phi(L_i)$ is semilinear for all i. Since any intersection of semilinear sets is semilinear, L would have a semilinear Parikh image. However, L does not have a semilinear Parikh image [30], a contradiction. □

There is a nice characterization of groups whose word problem is accepted by a BMM in [11], where it is shown that the word problem is accepted by such a machine with n counters if and only if the group G has a free abelian subgroup of rank n of finite index in G. Whilst the class of groups is the same as that characterized here, the languages accepted by BMMs form a

proper subclass of $\mathcal{PNL}$ (see [17]). The proof in [11] has some similarities with our approach.

We finish with a comment relating groups with a word problem in $\mathcal{PNL}$ to those with a word problem in $co\mathcal{CF}$. The latter is a very interesting class of groups (see [25, 37] for example) but we do not yet have a classification as to which groups lie in this class. However, we can say the following:

Proposition 145. *If $G \in \mathcal{PNL}$ then $G \in co\mathcal{CF}$, but the converse is false.*

Proof. By Proposition 6 in [25], all virtually abelian groups are in $co\mathcal{CF}$, and so the inclusion follows from Proposition 139.

For the converse consider the free group on two generators; this is not virtually abelian, and so is not in $\mathcal{PNL}$, but it is in $co\mathcal{CF}$ (see [25] for example). □

Let us try to summarize all the relations in a diagram. Recall from Section 3.1 our diagram of the inclusions among the various classes of formal languages. We now know that $\mathcal{PNL}$, the class of terminal Petri net languages, is incomparable to both $\mathcal{OC}$ and $\mathcal{CF}$. However, it properly contains the class of regular languages and is properly contained in the class of recursive languages. In contrast to this, the incomparability situations disappear when we are talking only of languages which are word problems of groups. Denoting these classes with subscripts G (i.e. OC_G is the class of languages in $\mathcal{OC}$ which are also word problems), the relationships now become:

$$\mathcal{REG}_G \subset \mathcal{OC}_G \subset PNL_G = \left(\bigcap_{\text{fin}} OC \right)_G \subset coCF_G \subset R_G \subset RE_G.$$

Witnesses for the strictness of the inclusions are, in order from left to right: the word problem of any group which is virtually cyclic but not finite ($\mathbb{Z}$, for example), the word problem of any group which is virtually abelian but not virtually cyclic ($\mathbb{Z} \times \mathbb{Z}$ for example), the word problem of the free group of rank two, the word problem of the Heisenberg group, and the word

problem of a group of type T (see Section 5.4). The incomparability between CF_G and PNL_G remains, however, since there are virtually abelian groups which are not virtually free and vice-versa. Their intersection is exactly the OC_G, as a group is both virtually abelian and virtually free if and only if it is virtually cyclic.

6.6 Standard form theorem

In this section we will show that if a language is the word problem of a virtually abelian group, then it can be recognized by a labelled Petri net in standard form - namely, one with the same initial and terminal marking, and we give a way to construct this Petri net. We will call this a *tidy* Petri net. This kind of language has been mentioned in [19] under the name cyclic terminal languages.

The first and easiest thing to note is that any abelian group is recognized by a tidy Petri net: infinite cyclic groups satisfy this property (see Figure 6.4.1 - the initial and terminal marking are both the empty marking) and so do finite cyclic groups (the corresponding Petri net is a cycle with a place for every possible exponent of the generator - the initial and terminal marking are then the marking with a token in the place for the 0th exponent and no tokens anywhere else). By the classification of finitely generated abelian groups, any abelian group's word problem will be recognized by a disjoint union of tidy Petri nets, where the initial/terminal marking is the "disjoint union" of initial/terminal markings of its components (what we mean by this is the operation that sends $((a, b), (c, d, e))$ to (a, b, c, d, e), rather more similar to concatenation of tuples).

THE CONSTRUCTION

Let G be a virtually abelian group, with abelian subgroup H of finite index. Say H is generated by the finite generating set Ω closed under inverses. Let $\{1, g_1, \ldots g_n\}$ be a set of coset representatives of H in G. Define

$$\Lambda := \{g_1, \ldots, g_n, g_1^{-1}, \ldots, g_n^{-1}\}$$

and $\Lambda^+ := \{g_1, \ldots, g_n\}$, $\Lambda^- := \{g_1^{-1}, \ldots, g_n^{-1}\}$.

We know that there exists a tidy labelled Petri net $P_H = (S, T, W, m_0, \Omega, l)$ recognizing the word problem of H with respect to Ω. From P_H, we will construct a new tidy labelled Petri net $P = (S', T', W', m_0', \Sigma, l')$ with underlying alphabet $\Sigma = \Omega \cup \Lambda$. This tidy Petri net will recognize the word problem of G with respect to Σ.

The set S' of places of P: We extend S with a place for every coset representative of H in G, including 1:

$$S' := S \cup \left\{p_g : g \in \Lambda^+ \cup \{1\}\right\}.$$

We will later see that a token in one of these places will indicate which coset of H a word we have read is in. We call this token the *coset marker*.

The initial/terminal marking m_0' of P: The initial marking will be identical to m_0 on S, and have a coset marker in the place p_1. So

$$\begin{aligned} &m_0'(p) = m_0(p) \qquad \text{for all } p \in S, \\ &m_0'(p_1) = 1, \\ &m_0'(p) = 0 \qquad \text{for all } p \in S' - (S \cup \{1\}). \end{aligned}$$

Transitions, arrows and labels of P: We shall give the definitions of T', W' and l' all at once in order to facilitate their description.

- Firstly, we stipulate that $T \subseteq T'$, $W' \restriction_{(T\times S)\cup(S\times T)} = W$, and $l' \restriction_T = l$, so that P is truly an extension of P_H.

- The place p_1 will be connected with a simple arrow to and from each transition in our original net P_H. It thus acts similarly to the 'run' place described in Section 6.2. So

$$W'(p_1, t) = W'(t, p_1) = 1$$

  for all $t \in T$. This is because we want P_H to be active only when the coset marker indicates that we are in H.

- Let $g \in \Lambda^+ \cup \{1\}$ be arbitrary. For each of these we will go through the process described below.

  Let $x \in \Sigma$. We know that $gx \in Hg'_{(g,x)}$ for some unique $g'_{(g,x)} \in \Lambda^+ \cup \{1\}$. So we choose a shortest word $\alpha_{(g,x)} \in \Omega^*$ such that

$$gx =_G \alpha_{(g,x)} g'_{(g,x)}.$$

  However, recall that many transitions in T may have the same label. There are therefore many sequences of transitions whose labels spell out $\alpha_{(g,x)}$. Each sequence σ may be represented by a vector $\Delta(\sigma) := C\Psi(\sigma)$ where C is the incidence matrix[3] of P_H and Ψ the Parikh mapping.

  In other words, let

$$E\left(\alpha_{(g,x)}\right) := \left\{\Delta(\sigma) | \sigma \in T^*, l(\sigma) = \alpha_{(g,x)}\right\}.$$

  Now for each vector $\Delta(\sigma) \in E\left(\alpha_{(g,x)}\right)$, add a new transition $t_{(g,x)}^{\Delta(\sigma)} \in T'$ such that

[3]The incidence matrix $C : T \times S \to \mathbb{Z}$ is a matrix such that $C(t, s) = W(t, s) - W(s, t)$. Thus $\Delta(\sigma)$ is essentially a vector in $\mathbb{Z}^{|S|}$ where each entry corresponding to $s \in S$ keeps track of the sum of what σ does to it - tokens added and tokens taken away.

– This transition is labelled by x:

$$l'\left(t^{\Delta(\sigma)}_{(g,x)}\right) = x,$$

– $t^{\Delta(\sigma)}_{(g,x)}$ takes the coset marker from p_g and deposits it in $p_{g'_{(g,x)}}$:

$$W'\left(p_g, t^{\Delta(\sigma)}_{g,x}\right) = 1 = W'\left(t^{\Delta(\sigma)}_{g,x}, p_{g'_{(g,x)}}\right),$$

– $t^{\Delta(\sigma)}_{(g,x)}$ does to the places in S what reading $\alpha_{(g,x)}$ would have done: for all $p \in S$,

$$\Delta(\sigma)(p) > 0 \Rightarrow \begin{cases} W'\left(p, t^{\Delta(\sigma)}_{(g,x)}\right) = \Delta(\sigma)(p) \\ W'\left(t^{\Delta(\sigma)}_{(g,x)}, p\right) = 0 \end{cases}$$

$$\Delta(\sigma)(p) < 0 \Rightarrow \begin{cases} W'\left(p, t^{\Delta(\sigma)}_{(g,x)}\right) = 0 \\ W'\left(t^{\Delta(\sigma)}_{(g,x)}, p\right) = \Delta(\sigma)(p) \end{cases}$$

and both are 0 otherwise.

Here we should point out a special case: if $g = 1$ and $x \in \Lambda^+$, then $\alpha_{(1,x)} = \lambda$ and $g'_{(1,x)} = x$. So in this case we just get one transition $t_{(1,x)}$, labelled x and taking the coset marker from p_1 to p_x.

Now given any word $w \in \Sigma^*$, we define the following algorithm to rewrite w: Look at the leftmost letter of w which is in Λ, say f. Call the letter to its right $x \in \Sigma$.

1. If f is the last letter of the word w, stop.
2. If $f \in \Lambda^+$, then rewrite fx to $\alpha_{(f,x)}g'_{(f,x)}$, and
3. If $f \in \Lambda^-$, then rewrite f to $\alpha_{(1,f)}g'_{(1,f)}$.

Repeat from the leftmost letter in Λ. Note that this process always terminates. Each iteration of (2) brings the leftmost element of Λ strictly closer to the end of the word, and (3) cannot happen twice in a row, as the g' produced is always in $\Lambda^+ \cup \{1\}$!

Furthermore this rewriting is unambiguous and terminates in a unique word of $\Omega^*(\Lambda \cup \{\lambda\})$, which we will denote $\bar{w}$.

We can now prove our lemma:

Lemma 146. *Reading a word $w \in \Sigma^*$ as input into P is the same (gives the same possible markings) as reading $\bar{w}$.*

Proof. We prove this by induction on the number k of rewrite steps from w to $\bar{w}$.

If $k = 0$, then $w = \bar{w} \in \Omega^*(\Lambda \cup \{\lambda\})$ and there is nothing to prove.

For the induction step, let w' be the result of the first rewrite step from w, i.e. $w \rightsquigarrow w' \rightsquigarrow^* \bar{w}$ (where $\rightsquigarrow^*$ denotes an arbitrary number of rewrite steps, including possibly none). By induction, reading w' gives the same markings as reading $\bar{w}$, so we only need to show that reading w is the same as reading w'.

Look at the leftmost letter in w of Λ - the one which will be rewritten in w', and the letter directly to its right. Call them f and x respectively. We can assume that

$$w = ufxv$$

where $u \in \Omega^*$,$f \in \Lambda$, $x \in \Sigma$ and $v \in \Sigma^*$. Let us observe what happens in P while reading w. Until the end of u, the only transitions fired are those in T - because of the coset marker in p_1, the only part of the net which is activated is the original net P_H. We now read f. There are two cases:

1. $f \in \Lambda^+$**.** Until we read f, the coset marker was still in p_1. Since f is already a coset representative, we just fire the transition $t_{(1,f)}$ labelled f taking the coset marker from p_1 and depositing it in p_f. The interesting

part happens when we read the next letter, x. Say

$$fx = \alpha_{(f,x)} g_i$$

where α is the shortest such word in Ω^* (picked in the definition of our tidy Petri net above). We know that there is at least one sequence of transitions in P_H reading $u\alpha_{(f,x)}$, because $u\alpha_{(f,x)}\alpha_{(f,x)}^{-1}u^{-1} \in W_\Omega(H)$. By definition, at least one of those is a sequence of transitions reading u followed by a sequence $\sigma \in T^*$ such that $\Delta(\sigma) \in E\left(\alpha_{(f,x)}\right)$ (in other words, at least one of the σs will be enabled after reading u).

After reading uf, we therefore fire the transition $t_{\alpha_{(f,x)}}^{\Delta(\sigma)}$ (with σ as above). By definition, this will take the coset marker from p_f and deposit it into p_{g_i}. Furthermore, its effect on the places S of P_H is exactly as if $\alpha_{(f,x)}$ had been read. Therefore overall we have read the equivalent of $u\alpha_{(f,x)}v$, which is exactly w'.

2. $f \in \Lambda^-$**.** Then we do exactly the same as above, but using $f = \alpha_{(1,f)} g_i$ only - we end up reading the equivalent of $u\alpha_{(1,f)}xv = w'$.

□

Because of the way we defined the rewriting rules, clearly $w =_G \bar{w}$, so one is in the word problem of G if and only if the other is. If a word w is in the word problem, since $\bar{w} \in \Omega^*(\Lambda \cup \{\lambda\})$, it must be the case that $\bar{w} = uv$ where u is in the word problem of H and $v = \lambda$. By Lemma 146 above, this means that we read u in such a way that we are back at the initial marking of P_H (because P_H is tidy), and the coset marker is back in p_1 where it was in the beginning. Therefore w is accepted by P.

Conversely, assume that w is a word accepted by the Petri net P - we need to show w is in the word problem of G.

Lemma 147. *Any sequence of transitions accepted by the Petri net P represents a word in the word problem of G.*

Proof. Let τ be a sequence of transitions leading from m'_0 back to m'_0. We want to show that $w := l(\tau) =_G 1_G$.

First, note that to simplify things we can assume H to be normal[4] in G. The consequence of this is that the only transitions within τ that will change the place of the coset marker are transitions of type $t^{\Delta(\sigma)}_{(g,f)}$ where g and f are both in $\Lambda \cup \{1\}$ (note that g is always in $\Lambda^+ \cup \{1\}$). Each of these transitions takes the coset marker from p_g to the place for whatever coset the group element gf is in. Therefore, each of these transitions equates to multiplying the previous coset representative on the right with f (call f the 'second component' of $t^{\Delta(\sigma)}_{(g,f)}$).

Now look at the subsequence τ' of τ consisting only of transitions $t^{\Delta(\sigma)}_{(g,f)}$ as above. The coset place the marker ends up in will be the coset where the product of all their second components, in order, is. However, at the end of τ the coset marker must be back in p_1 (it also starts out in p_1), so the product of all second components of τ' must represent 1_G. Luckily, the product of all second components of τ' is also the label $l(\tau')$! So $l(\tau') =_G 1_G$.

Let us come back to τ and w for a moment. By construction, we know that there is a sequence ρ of transitions which is τ with all $t^{\Delta(\sigma)}_{(f,x)}$ replaced by σ. This is essentially a sequence of transitions corresponding to the rewritten version $\bar{w}$ of w - by the comment above about the overall effect of transitions of type $t^{\Delta(\sigma)}_{(g,f)}$, we can ignore their effect on the coset places and only take into account their effect on the places of P_H. Now by construction, the sequence of markings defined by ρ on P_H is the same as that defined by τ. Since τ was accepted by P, so must ρ be. Since ρ takes place solely in P_H, it goes from m_0 to m_0, so $1 =_G l(\rho) = \bar{w} =_G w$ and we are done. □

This doesn't hold for Petri net languages which are not word problems of groups, as it implies that the language must be closed under concatenation

[4]Replace H by its normal core, which also has to have finite index in G.

and indeed Kleene star. Clearly the language

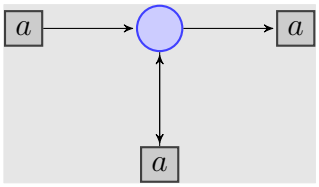
$$L = \{a^n b^m : 1 \leqslant m \leqslant 2^n, 1 \leqslant n\}$$

(see Proposition 144) does not have this property.

6.6.1 A comment on regular languages

Inspired by the discussion above, we shortly remark here that the same sort of thing is not true for regular languages: we can have a regular language which is closed under Kleene star but which is not accepted by any DFA with the same start and final state. As an example, consider $L = \{\lambda\} \cup \{a^n : n \geqslant 2\}$. In this case clearly $L = L^*$, but it is straightforward to see that there is no tidy DFA recognizing it - this can be proven by means of minimal DFAs defined below.

Note that there is, however, a tidy Petri net accepting L, even though L is not the word problem of a group (in order to be so, it would have to have property (ii) in Theorem 149 below). This means that non-word problem terminal Petri net languages can also occasionally have the desired property. The tidy Petri net recognizing L is shown below. The initial and terminal marking is the empty marking.



However, if we add to $L = L^*$ the condition

$$(\alpha\beta \in L) \wedge (\alpha \in L) \Rightarrow (\beta \in L)$$

then we do have a tidy DFA recognizing L. To show this we need to introduce

the concept of a minimal DFA.

Definition 148. *Recall that a deterministic finite automaton*

$$\mathfrak{A} = (Q, \Sigma, \delta, s, F)$$

is simply one where the transition relation δ is a function. As usual we extend this function to words, giving a function $Q \times \Sigma^* \to Q$.

For $p, q \in Q$ two states, we say that p and q are distinguishable *if there is $w \in \Sigma^*$ such that $\delta(p, w) \in F$ but $\delta(q, w) \notin F$.*

For a language $L \subseteq \Sigma^*$, $\mathfrak{A}$ is a minimal DFA *recognizing L if it has no indistinguishable states and no unreachable states.*

It is a consequence of the Myhill-Nerode theorem that any regular language has a minimal DFA recognizing it (it in fact has a minimum state one - see page 67 of [26]).

We now prove our claim:

Theorem 149. *If L is a regular language such that*

i. $L = L^*$ *and*

ii. $(\alpha\beta \in L) \wedge (\alpha \in L) \Rightarrow (\beta \in L)$,

then L is recognized by a DFA with the same initial and terminal state.

Proof. Let $\mathfrak{A} = (Q, \Sigma, \delta, s, F)$ be a minimal DFA recognizing L. Since $\lambda \in L$ (because $L^* = L$), the start state must be accepting. Assume for a contradiction that there is another accept state, f. We will show that s and f are indistinguishable, contradicting the minimality of $\mathfrak{A}$.

First assume $\delta(s, w) \in F$ for some $w \in \Sigma^*$. We will show that $\delta(f, w) \in F$. We know that there is a $u \in \Sigma^*$ such that $\delta(s, u) = f$, otherwise f is just an unreachable state and we can discard it. We know that $uw \in L$ by (i). Since $\mathfrak{A}$ is deterministic, there must be a path reading w from f to a final state. In other words, $\delta(f, w) \in F$.

Conversely, assume $\delta(f, w) \in F$. Let u be as above. Then uw is accepted by the automaton and hence is in L, and also $u \in L$ by definition. But then by (ii), $w \in L$ which means exactly that $\delta(s, w) \in F$ and we are done. $\square$

Note that our two conditions above are strictly weaker than L being a word problem of a group: in [48], Parkes and Thomas show that the following two properties characterize languages $L \subseteq \Sigma^*$ which are word problems of groups generated by the alphabet Σ:

1. For all $\alpha \in \Sigma^*$ there is $\beta \in \Sigma^*$ such that $\alpha\beta \in L$, and

2. If $\alpha\gamma\beta \in L$ and $\gamma \in L$ then $\alpha\beta \in L$.

An example of a language which satisfies (i) and (ii) of the previous theorem but not (1) above is the language $L = \{(ab)^n : n \in \mathbb{N}\}$ - any word that starts with a b is impossible to complete to a word in L.

It would be nice to know whether the two conditions (i) and (ii) above are enough for $\mathcal{PNL}$ languages to have normal form Petri nets as well, but unfortunately we don't know how to do it without our correspondence theorem.

Chapter 7

Conclusion and future work

In this thesis we have presented a fairly eclectic mix of new results, and attempted to give the background work they are building on. The thread tying all of the different questions in this thesis is the deduction of algebraic properties of groups from language-theoretic facts about their word problems and variations.

In our fourth chapter we have gathered and sometimes generalized previous work on word problems of pairs of groups, their closure properties and algebraic relationships. It is clear that much more work would be possible on this subject - our conjecture is still not proven, and the surface of the connection with normal cores of subgroups has barely been scratched. Especially on the subject of one-counter pairs of groups, the main problem is that we do not know of many examples of one-counter pairs where the large group is not already one-counter. This makes both the formulating of hypotheses and the hunt for counterexamples difficult.

In the fifth chapter our main contribution is the result that groups where all irreducible word problems are recursively enumerable are actually groups with solvable word problem. We add to this a partial classification of partially mimsy groups: those with some irreducible word problems which are recursively enumerable and others which are not, if they do indeed exist. A

clear future aim here is to either find a partially mimsy group or show that none can exist. If there are partially mimsy groups, in order to have a full classification, we would need to prove some sort of converse to Theorem 114. We do know that the free product of a partially mimsy group and a uniformly mimsy group must be partially mimsy. However, in Theorem 114 the free product is constructed with uniformly mimsy and non-mimsy groups. Therefore we would need to know what properties the free product of a uniformly mimsy and a uniformly non-mimsy group has, or the free product of a universally non-mimsy and a partially mimsy group.

The sixth chapter concerns terminal Petri net languages, and our main contribution is that the groups whose word problems are such languages are exactly the virtually abelian groups. In our opinion this last chapter is the richest with regards to possibilities for future work. We mentioned investigating whether we still obtain the virtually abelian groups if we allow our Petri nets to have λ-transitions. This would be an interesting result, as this class of languages (which we call $\mathcal{PNL}^{\lambda}$) is in fact closed under arbitrary homomorphisms rather than only λ-free homomorphisms. Its closure properties are therefore more helpful and more comparisons with other language classes may be possible. It seems as though all parts of our main result for $\mathcal{PNL}$ could be replicated for $\mathcal{PNL}^{\lambda}$, with the exception of the growth of groups. It is not clear to us that groups with $\mathcal{PNL}^{\lambda}$ word problems have polynomial growth - the possibility of padding words with arbitrary numbers of λs poses a problem.

Other additional questions about $\mathcal{PNL}$ could be asked. One of the more interesting ones is as follows: we know that the equivalence problem for Petri nets is undecidable (see Section 8.2 in [19]), meaning that given two Petri nets it is undecidable whether their terminal languages are the same. However, as we have seen the intersection of Petri net languages with word problems of groups is rather better behaved than the whole class - could it be that given two Petri nets which we *know* recognize word problems of groups, we could

decide equivalence? The idea here is to use our classification, and to in fact deduce a group presentation from the Petri net. Since it is decidable amongst virtually abelian groups whether two presentations give the same group (in fact, this is true for a larger class, that of the virtually polycyclic groups - see [55]), we would be done. In order to try to guess a presentation from a Petri net known to recognize a word problem, we would try to enumerate partitions on the set of generators (and hence the alphabet of the Petri net) into two sets, one of which generates the abelian group of finite index and the other contains coset representatives. The difficult part here is to deduce a presentation for the abelian part.

There are a plethora of decidability questions about PNLs which we haven't had time to investigate in this thesis, such as: Given a PNL, is it decidable whether it is the word problem of a group? Given a PNL which is known to be the word problem of a group, is it decidable whether it is regular? One-counter? Context-free? Whether it has semilinear Parikh image?

In sum, this thesis is but a small fishing hole on the surface of a frozen lake, but we hope to melt some more of the ice in the future.

Bibliography

[1] Anisimov, A.: Group languages. Cybernetics and Systems Analysis **7** (1971) 594–601 10.1007/BF01071030.

[2] Anisimov, A.: Some algorithmic problems for groups and context-free languages. Cybernetics **8**(2) (1972) 174–182

[3] Boone, W.W.: The word problem. Annals of Mathematics **70** (1959) 207–265

[4] Boone, W.W., Higman, G.: An algebraic characterization of groups with a solvable word problem. Journal of the Australian Mathematical Society **18** (1974) 41–53

[5] Ceccherini-Silberstein, T., Woess, W.: Context-free pairs of groups i: Context-free pairs and graphs. European Journal of Combinatorics **33**(7) (2012) 1449 – 1466 Groups, Graphs, and Languages.

[6] Chrzastowski-Wachtel, P.: Testing undecidability of the reachability in petri nets with the help of 10th hilbert problem. In Donatelli, S., Kleijn, J., eds.: Application and Theory of Petri Nets 1999. Volume 1639 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1999) 268–281

[7] Collins, D., Zieschang, H.: Combinatorial Group Theory and Fundamental Groups. Institut des Hautes Etudes Scientifiques (1989)

[8] Droms, C., Servatius, B., Servatius, H.: Connectivity and planarity of Cayley graphs. Beiträge zur Algebra und Geometrie **39** (1998) 269–282

[9] Dunwoody, M.J.: The accessibility of finitely presented groups. Inventiones Mathematicae **81** (1985) 449–457

[10] Dunwoody, M.: An inaccessible group. In Niblo, G.A., Roller, M.A., eds.: Geometric Group Theory. Volume 1 of London Mathematical Society Lecture Notes Series 181., Cambridge University Press (1993) 75–78

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

[12] Fonseca, A.R.: Formal languages and the irreducible word problem in groups. PhD thesis, University of Leicester (2005)

[13] Fonseca, A.R., Parkes, D.W., Thomas, R.M.: Irreducible word problems in groups. In Campbell, C.M., Quick, M.R., Robertson, E.F., Smith, G.C., eds.: Groups St Andrews 2005 Volume 1. LMS Lecture Notes Series 339, Cambridge University Press (2007) 327–340

[14] Fonseca, A.R., Thomas, R.M.: Context-free irreducible word problems in groups. In Fine, B., Gaglione, A.M., Spellman, D., eds.: Combinatorial Group Theory, Discrete Groups, and Number Theory. Contemporary Mathematics 421, American Mathematical Society (2006) 125–136

[15] Freudenthal, H.: Über die Enden diskreter Räume und Gruppen. Comment. Math. Helv. **17** (1945) 1–38

[16] Golod, E.S., Šafarevič, I.R.: On the class field tower. Izv. Akad. Nauk SSSR Ser. Mat. **28** (1964) 261–272

[17] Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science **7** (1978) 311–324

[18] Gromov, M.: Groups of polynomial growth and expanding maps. Publications Mathematiques de l'Institut des Hautes Etudes Scientifiques **53**(1) (1981) 53–78

[19] Hack, M.: Petri net languages. Computation Structures Group Memo 124, Project MAC, M.I.T. (1975)

[20] Haring-Smith, R.H.: Groups and simple languages. Transactions of the American Mathematical Society **279** (1983) 337–356

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

[22] Herbst, T., Thomas, R.M.: Group presentations, formal languages and characterizations of one-counter groups. Theoretical Computer Science **112** (1993) 187–213

[23] Higman, G.: Subgroups of finitely presented groups. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **262** (1961) 455–475

[24] Holt, D.F., Owens, M.D., Thomas, R.M.: Groups and semigroups with a one-counter word problem. Journal of the Australian Mathematical Society **85**(02) (2008) 197–209

[25] Holt, D.F., Rees, S., Röver, C.E., Thomas, R.M.: Groups with context-free co-word problem. Journal of the London Mathematical Society **Vol.71**(No.3) (June 2005) 643–657

[26] Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley (1979)

[27] Hopf, H.: Enden offener rume und unendliche diskontinuierliche gruppen. Commentarii Mathematici Helvetici **16**(1) (1943) 81–100

[28] Houghton, C.: Ends of locally compact groups and their coset spaces. Journal of the Australian Mathematical Society **17**(03) (1974) 274–284

[29] Ito, M., Kari, L., Thierrin, G.: Insertion and deletion closure of languages. Theoretical Computer Science **183** (1997) 3–19

[30] Jantzen, M.: On the hierarchy of petri net languages. Informatique Theorique et Applications **13**(1) (1979)

[31] Jantzen, M.: Language theory of petri nets. In Brauer, W., Reisig, W., Rozenberg, G., eds.: Petri Nets: Central Models and Their Properties. Volume 254 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1987) 397–412

[32] Johnson, D.: Presentations of Groups. London Mathematical Society Student Texts. Cambridge University Press (1997)

[33] Kapovich, I., Myasnikov, A.: Stallings foldings and subgroups of free groups. Journal of Algebra **248**(2) (2002) 608–668

[34] Kari, L.: Insertion operations: Closure properties. Bulletin of the European Association for Theoretical Computer Science **51** (1993) 181–191

[35] Lakin, S.R., Thomas, R.M.: Context-sensitive decision problems in groups. In Calude, C.S., Calude, E., Dinneen, M.J., eds.: Developments in Language Theory: 8^{th} International Conference. Lecture Notes in Computer Science 3340, Springer-Verlag (2004) 296–307

[36] Lambert, J.: A structure to decide reachability in petri nets. Theoretical Computer Science **99**(1) (1992) 79 – 104

[37] Lehnert, J., Schweitzer, P.: The co-word problem for the Higman-Thompson group is context-free. Bulletin of the London Mathematical Society **39** (2007) 235–241

[38] Lyndon, R.C., Schupp, P.E.: Combinatorial group theory. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag (1977)

[39] Madlener, K., Otto, F.: About the descriptive power of certain classes of finite string-rewriting systems. Theoretical Computer Science **67** (1989) 143–172

[40] Mayr, E.W.: An algorithm for the general petri net reachability problem. SIAM Journal on Computing **13**(3) (1984) 441–460

[41] Meier, J.: Groups, Graphs, and Trees : An Introduction to the Geometry of Infinite Groups. London Mathematical Society Student Texts. Cambridge University Press, Cambridge (2008)

[42] Muller, D.E., Schupp, P.E.: The theory of ends, pushdown automata, and second-order logic. Theoretical Computer Science **37** (1985) 51 – 75

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

[44] Novikov, P.S.: On the algorithmic unsolvability of the word problem in group theory. Trudy Matematicheskogo Instituta imeni VA Steklova **44** (1955) 1–143

[45] Ol'shanskii, A.: Groups of bounded period with subgroups of prime order. Algebra and Logic **21**(5) (1982) 369–418

[46] Parikh, R.J.: On context-free languages. Journal of the ACM **13**(4) (October 1966) 570–581

[47] Parkes, D.W., Shavrukov, V.Y., Thomas, R.M.: Monoid presentations of groups by finite special string-rewriting systems. Theoretical Informatics and Applications **38** (2004) 245–256

[48] Parkes, D.W., Thomas, R.M.: Groups with context-free reduced word problem. COmmunications in Algebra **30** (2002) 3143–3156

[49] Petersen, J.L.: Computation sequence sets. Journal of Computer and System Sciences **13**(1) (1976) 1 – 24

[50] Rino Nesin, G., Thomas, R.: Groups with a recursively enumerable irreducible word problem. In Gasieniec, L., Wolter, F., eds.: Fundamentals of Computation Theory. Volume 8070 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 283–292

[51] Rino Nesin, G., Thomas, R.: Groups whose word problem is a Petri net language. In Shallit, J., Okhotin, A., eds.: Descriptional Complexity of Formal Systems. Volume 9118 of Lecture Notes in Computer Science. Springer International Publishing (2015) 243–255

[52] Robinson, D.: A Course in the Theory of Groups. 2nd edn. Springer (1995)

[53] Röver, C.E.: On groups which are syntactic monoids of deterministic context-free languages. International Journal of Algebra and Computation **14**(04) (2004) 499–504

[54] Scott, P.: Ends of pairs of groups. Journal of Pure and Applied Algebra **11**(13) (1977) 179 – 198

[55] Segal, D.: Decidable properties of polycyclic groups. Proc. London Math. Soc **3** (1990) 61–497

[56] Stallings, J.R.: Group theory and three-dimensional manifolds. Yale University Press (1971)