TODD-COXETER METHODS FOR INVERSE MONOIDS

Andrew Cutting

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews

2001

Todd-Coxeter Methods for Inverse Monoids

By Andrew Cutting

A thesis submitted for the degree of Doctor of Philosophy of the University of St. Andrews.

TL
D786

Declarations

I Andrew George Cutting hereby certify that this thesis has been composed by myself, that it is a record of my own work, and that it has not been accepted in partial or complete fulfillment of any other degree of professional qualification.

Signed .... Date ..12/3/01.

I was admitted to the Faculty of Science of the University of St Andrews under Ordinance General No12 on October 1996 and as a candidate for the degree of Ph.D. on October 1997.

Signed .... Date ..12/3/01..

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be supplied to any bona fide library or research worker.

Signed .... Date .12/3/01

I hereby certify that the candidate has fulfilled the conditions of the Resolutions and Regulations appropriate to the degree of Ph.D.

Signature of Supervisor .... Date 12/3/01...

Acknowledgements

I would like to express my gratitude to my supervisors Edmund Robertson and Mike Atkinson and also to Werner Nickel, Nikola Ruškuc, Andrew Solomon, Steve Linton, Robert Wainwright, Hayrullah Ayik, Robert Arthur, Brunnetto Piochi and Allessandra Cherubini who helped me with so many mathematical ideas.

I would like to thank E. P. S. R. C. for the financial support.

I would like to thank my fiance Hilary Anderson and my family for all their patience.

Abstract

Let P be the inverse monoid presentation $\langle X|U\rangle$ for the inverse monoid M, let π be the set of generators for a right congruence on M and let $u \in M$. Using the work of J. Stephen [15], the current work demonstrates a coset enumeration technique for the $\mathcal{R}$-class R_u similar to the coset enumeration algorithm developed by J. A. Todd and H. S. M. Coxeter for groups. Furthermore it is demonstrated how to test whether $R_u = R_v$ for $u, v \in M$ and so a technique for enumerating inverse monoids is described. This technique is generalised to enumerate the $\mathcal{H}$-classes of M.

The algorithms have been implemented in GAP 3.4.4 [25], and have been used to analyse some examples given in **Chapter 6**. The thesis concludes by a related discussion of normal forms and automaticity of free inverse semigroups.

Contents

Chapter 0

Introduction

In 1936, J. A. Todd and H. S. Coxeter [29] developed a systematic method for enumerating the cosets of a subgroup of a finitely presented group. This is one of the most important and usefull procedures in computational group theory and provides a vital link between group presentations and permutations. It was one of the first group theoretic algorithms to be implemented on a digital computer by C. B. Hazelgrove in 1953.

Although the algorithm is for groups it also generalises quite naturally to monoids (see B. H. Neumann [17] and A. Jura [12]). The current work describes an implementation of a Todd-Coxeter style algorithm for inverse monoids based on the work of J. Stephen [27]. There are, however, important differences in this procedure to the previous procedures such as the fact that the inverse monoid is divided into subsets which are enumerated seperately rather enumerating the entire structure in one go and also that in each of these subsets the enumeration terminates before the coset table is filled.

There is a set of relations called *Greens relations* on a monoid, these include $\mathcal{R}$-classes and $\mathcal{H}$-classes. In particular in an inverse monoid, every $\mathcal{R}$-class contains exactly one *idempotent*, e. The property of $\mathcal{R}$-classes in inverse monoids is that if $uu^{-1}v\mathcal{R}uu^{-1}$ then $uu^{-1}vv^{-1} = uu^{-1}$. It could be said that inverse monoid $\mathcal{R}$-classes have a kind of local right cancelation which is similar to groups. $\mathcal{H}$ is a subset of $\mathcal{R}$ such that there is both a right cancelation and a left cancelation in each $\mathcal{H}$-class. The reader is encouraged to keep group theoretic results in mind as we generalise them to inverse monoid $\mathcal{R}$-classes. See, for example, Howie [9] or Petrich [18] for basics on Green's relations.

The inverse monoid enumeration procedure is of necessity split into two parts - the enumeration of each $\mathcal{R}$-class and the enumeration of the $\mathcal{R}$-classes. The enumeration procedure for groups and monoids is only reminscent of the first of these parts. This splitting is, I believe, necessary and is related to the fact that the free inverse monoid over X cannot be finitely presented as a monoid. In essence we are dealing with two superimposed congruences - the one generated by an infinite number of relations for the monoid presentation of the free inverse monoid and the one generated by the extra relations from the inverse monoid presentation.

This work is primarily of use for those who wish to study inverse monoid presentations. It allows a detailed investigation into each of the $\mathcal{R}$-classes of an inverse monoid, with the only finiteness conditon being that each $\mathcal{R}$-class contains a finite number of $\mathcal{H}$-classes. It may also be of use for developing a technique to enumerate any algebraic structure whose word problem is solvable.

Chapter 1 is an introduction to Todd-Coxeter coset enumeration and details the working of the algorithm for monoids. This algorithm is based on Neumann's algorithm although I do not give an exact replica. Monoids are very general algebraic structures and it is partly my goal to understand the application of the coset enumeration *method* in its most general form.

Chapter 2 is an introduction to inverse monoid theory. Naturally the emphasis is on the computational and presentation theory side of inverse monoid theory.

Chapter 3 details W. D. Munn's work [16] on word trees and the solution of the word problem in the free inverse monoid. I then go on to apply Munn's ideas to construct an inverse monoid enumerator. The purpose of this chapter is entirely for the sake of explaining and exploring my reasoning about inverse monoid enumeration. The algorithm in **Section 3.4** is inferior in several respects to the algorithm in **Chapter 5**. **Chapter 3** can be skipped when reading this thesis.

Chapter 4 details Stephen's work [27] on the solution of word problem for general inverse monoids. I add a slight generalisation to do with right congruences.

Chapter 5 details the inverse monoid coset enumeration algorithm proper. I provide a proof that it terminates and produces the correct result and I also detail some variations for enumerating individual $\mathcal{R}$-classes and right quotients of $\mathcal{R}$-classes and a variation which enumerates $M/\mathcal{H}$.

In **Chapter 6**, using insights from the enumeration algorithm, I look at various types of inverse monoid presentation which include presentations for monogenic inverse monoids, coxeter inverse semigroups, symmetric inverse semigroups, free inverse semigroup products of finite inverse semigroups with semilattices, inverse semigroups with infinite $\mathcal{R}$-classes and inverse semigroups with an infinite $\mathcal{R}$-class.

Chapter 7 Contains a paper I wrote with Andrew Solomon concerning the automaticity of free inverse semigroups.

The implementation has been done in GAP 3.4.4 [25] and is included in the **Appendix**.

Chapter 1

Preliminaries

In this chapter I introduce some of the basic ideas involving groups and semigroups. In particular I am interested in free groups and semigroups, words in these structures, presentations of groups and semigroups and ultimately Todd-Coxeter coset enumeration.

1.1 Free Semigroups, Monoids and Groups

It is worth recalling some basic definitions before we proceed.

As I am interested in insights into enumeration techniques in the most general terms I shall talk about *algebraic structures*. By these I shall mean a set, A, with certain *operations*. An n-ary operation (with $n \geq 1$) being a mapping from the Cartesian product of n copies of A into A. If $n = 0$ then this *nullary operation* is a simply a specific element in A. Almost all algebraic structures that mathematicians are interested in only involve binary, unary and nullary operations. In particular if $*$ is a binary operation on A and $x, y \in A$ then the image of (x, y) under $*$ is written *multiplicatively* as $x * y$. I shall talk about certain standard notions such as *homomorphisms* and *substructures* and would refer the reader to a standard algebra textbook such as Burris and Sankappanavar [2].

Definition 1.1.1. A *semigroup*, S, is a set with a binary operation $*$ such that $*$ is associative that is

G1 $x * (y * z) = (x * y) * z, \forall x, y, z \in S.$

A *monoid*, M, is a semigroup with an identity $\epsilon_M \in M$ such that

G2 $x * \epsilon_M = \epsilon_M * x = x \,, \forall x \in M.$

A *group*, G, is a monoid with inverses, that is it has a unary operation $^{-1}$ such that

G3 $x * x^{-1} = x^{-1} * x = \epsilon_G, \; \forall x \in G.$

Where there is no confusion we write xy instead of $x * y$ and we write ϵ in instead of ϵ_M. The equations in the axioms G1, G2 and G3 are called *identities*.

Definition 1.1.2. A *variety*, $\mathcal{V}$, is a collection of algebraic structures with the following characteristics:

V1 $\mathcal{V}$ is closed under homomorphisms. That is if $O \in \mathcal{V}$ and O' is a homomorphic image of O then $O' \in \mathcal{V}$.

V2 $\mathcal{V}$ is closed under taking of substructures. That is if O' is a substructure of $O \in \mathcal{V}$ then $O' \in \mathcal{V}$.

V3 $\mathcal{V}$ is closed under taking direct products. That is if $\{O_i | i \in I\} \subseteq \mathcal{V}$ then the Cartesian product $\prod_{i \in I} O_i \in \mathcal{V}$.

The collection of semigroups, the collection of monoids and the collection of groups are varieties. We call elements (eg. single groups, single semigroups etc.) of a variety *objects*.

NOTATION: I shall refer to $\mathcal{S}$ as the variety of semigroups, $\mathcal{M}$ as the variety of monoids and $\mathcal{G}$ as the variety of groups.

Definition 1.1.3. Given a variety $\mathcal{V}$ and a set X, then an object, F is said to be *free over X in $\mathcal{V}$* if $X \subseteq F$ and for every object $O \in \mathcal{V}$ and any mapping $\phi : X \to O$ there is a unique homomorphism $\phi' : F \to O$ which extends ϕ ie. $x\phi = x\phi'$ for $x \in X$.

In particular we have *free groups*, *free semigroups* and *free monoids*.

We have the following well know lemma. See for example Burris and Sankappanavar [2] for a proof.

Lemma 1.1.4. *Given a variety $\mathcal{V}$ and a set X then the free object over X in $\mathcal{V}$ exists and is unique.*

NOTATION: The free object over X in variety $\mathcal{V}$ is denoted by $\mathbf{F}_{\mathcal{V}}(X)$.

The definition of a free object is quite abstract so I shall introduce some notions to help "concretise" them for semigroups, monoids and groups.

Definition 1.1.5. Let X be a set. A *word* over X is a string of elements of X. The elements of X are refered to as *letters*. We call the string of zero length the *empty word* and denote it by ϵ. If $w = x_1x_2...x_n$ is a word then all the words of the form $x_ix_{i+1}...x_j$ $(1 \leq i \leq j \leq n)$ are all *subwords of* w. The set of all words (including the empty word) over X is denoted by X^*. We denote $X^*\backslash\{\epsilon\}$ by X^+.

If we define a binary operation, $*$, on X^* and X^+ by concatenation ie.

$$(x_1x_2...x_n) * (y_1y_2...y_m) = x_1x_2...x_ny_1y_2...y_m$$

where $x_1, x_2, ..., x_n, y_1, y_2, ..., y_m \in X$ then it is easy to check that X^* is a monoid with the empty word as the identity, and that X^+ is a semigroup.

Given a monoid M and a set X with a map $\phi : X \to M$, then we define $\phi' : X^* \to M$ by

$$(x_1x_2...x_n)\phi' = (x_1\phi)(x_2\phi)...(x_n\phi)$$

and

$$\epsilon\phi' = \epsilon_M.$$

It is easy to see that ϕ' is a homomorphism which extends ϕ. Moreover by the definition of homomorphism $(uv)\theta = (u\theta)(v\theta)$ for any homomorphism $\theta : X^* \to M$ and $\epsilon\theta = \epsilon_M$ and so we see that ϕ is unique. Hence X^* is the free monoid over X. Similarly X^+ is the free semigroup over X.

In a general setting a *congruence* is an equivalence relation on a structure which is consistent with operations and relations of that structure. By consistency there is a similar kind of structure on the set of equivalence classes, or *congruence classes*, which is called a *quotient*. For a semigroup (or monoid) S a congruence ρ is consistent with multiplication ie. if $x\rho y$ then $xz\rho yz$ and $zx\rho zy$ for $x, y, z \in S$. It turns out that if an equivalence relation on a group is consistent with multiplication then it is also consistent with taking inverses and is thus a group congruence. If S is a semigroup (or a monoid or a group) and ρ is just a congruence on S then we write the *quotient semigroup* (or *quotient monoid* or *quotient group*) as S/ρ.

A *right congruence* ρ on a semigroup/monoid/group S is consistent with "right multiplication" ie. if $x\rho y$ then $xz\rho yz$. We also have the dual *left congruence* so that a congruence is both a left congruence and a right congruence. The set of equivalence classes of a right congruence ρ on a semigroup/monoid/group only form a new semigroup/monoid/group when ρ is a congruence.

It would be no exaggeration to say that this thesis is about congruences and right congruences on semigroups. In particular we may define X^* as being isomorphic to a congruence ρ of $(X \cup \{\epsilon\})^+$ by using axiom G2. If we define a relation on $(X \cup \{\epsilon\})^+$

$$\eta = \{(x\epsilon, \epsilon x), (x\epsilon, x) | x \in X\},$$

then we can construct ρ by finding the intersection of all equivalence relations, eta_i, $i \in I$ on $(X \cup \{\epsilon\})^+$ which contain η such that for each $(u, v) \in \eta_i$ and for every $s, t \in (X \cup \{\epsilon\})^+$ then $(sut, svt) \in \eta_i$ $(i \in I)$. Now this is a convoluted way of saying, "you may cross out the ϵ's in any word in $(X^+ \cup \{\epsilon\})^+$," but it is the kind of construction we will be looking at.

We now turn our attention to the free group on X. Let X^{-1} be a set of symbols with the same cardinality as X such that $X \cap X^{-1} = \emptyset$. We define a bijection $^{-1} : X \to X^{-1}$ so that for $x \in X$, x^{-1} is the image of x under $^{-1}$. We extend $^{-1}$ to bijection of $(X \cup X^{-1})^*$ onto itself so that $(x^{-1})^{-1} = x$ for each $x \in X \cup X^{-1}$ and

$$(x_1 x_2 ... x_n)^{-1} = x_n^{-1} x_{n-1}^{-1} ... x_1^{-1}.$$

We call $^{-1}$ an *involution*. It may be characterised by the following identities:

I1 $(x^{-1})^{-1} = x$

I2 $(xy)^{-1} = y^{-1}x^{-1}$

Define a congruence σ on $(X \cup X^{-1})^*$ (or $\rho \cup \sigma$ on $(X \cup X^{-1} \cup \{\epsilon\})^+$ where ρ is the free monoid congruence) by first defining the relation

$$\zeta = \{(xx^{-1}, x^{-1}x), (xx^{-1}, \epsilon) | x \in X \cup X^{-1}\} \cup \{(u, u) | u \in (X \cup X^{-1})^*\}$$

on $(X \cup X^{-1})^*$. Define σ to be the intersection of all equivalence relations ζ_i, $i \in I$ containing ζ such that if $u, v \in \zeta_i$ and $s, t \in (X \cup X^{-1})^*$ then $(sut, svt) \in \zeta_i$.

It is a non-trivial fact that $(X \cup X^{-1})^*/\sigma$ is isomorphic to the free group over X because it turns out that σ is identical to

$$\{(xx^{-1}, x^{-1}x), (xx^{-1}, \epsilon) | x \in (X \cup X^{-1})^*\}$$

which is the intersection of all congruences which satisfy the identities in axiom G3 (see for example D.L. Johnson [11] for a proof of this).

Given a word $w \in (X \cup X^{-1})^*$, we define $\overline{w}$ to be the unique word in $w\sigma$ such that there are no subwords which contain xx^{-1} or $x^{-1}x$. We say that $\overline{w}$ is *freely reduced*and $\overline{w}$ is the *free reduction* of w. The free group on X is the set of freely reduced words. The multiplication is defined by concatenation followed by free reduction.

I shall summarise. We are interested in varieties which are subclasses of the variety of semigroups. Each variety, $\mathcal{V}$, has a unique free object for any given set X. The free object is defined as being isomorphic to the quotient of X^+ by the congruence which is defined as the intersection of all congruences ρ on X^+ such that $X^+/\rho \in \mathcal{V}$. In the cases of monoids and groups this intersection of congruences can be defined as a finite (if X is finite) relation which generates the congruence. This last fact is very convenient and is not true of other varieties such as inverse semigroups and completely regular semigroups.

1.2 Presentations

Consider a variety $\mathcal{V} \subseteq \mathcal{S}$. Any object $O \in \mathcal{V}$ is a homorphic image of $\mathbf{F}_{\mathcal{V}}(X)$ for some set X. Equivalently O is isomorphic to $\mathbf{F}_{\mathcal{V}}(X)/\rho$ for some set X and some congruence ρ. It is therefore natural to regard an object as a set of *generators*, X and a congruence on $\mathbf{F}_{\mathcal{V}}(X)$.

Definition 1.2.1. A *semigroup presentation* P is the pair of a set X and a relation U on X^+. It is written $\langle X|U\rangle$ and $(u, v) \in U$ is often written $u = v$. Here X is called the *set of generators*, while U is called the *set of relations*. Similarly a *monoid presentation* $P = \langle X|U\rangle$ is the pair of a set X and a relation U on X^*. A *group presentation* $P = \langle X|U\rangle$ is the pair of a set X and a relation U on $(X \cup X^{-1})^*$.

In the most general terms, given a variety $\mathcal{V}$, then a $\mathcal{V}$ presentation is the pair $P = \langle X|U\rangle$. The (unique) object defined by P is $\mathbf{F}_{\mathcal{V}}(X)/\rho$ where ρ is the intersection of all congruences which contain U. It not immediately obvious although it is the case that ρ is itself a congruence. Where it does not confuse anything I shall abuse the notation and write $u = v$ instead of $u\rho v$ or $x\rho = y\rho$. I shall refer to equality in the free semigroup and the free monoid as "$\equiv$" so as to avoid confusion.

EXAMPLE: The semigroup defined by the semigroup presentation $\langle x | x^4 = x^2 \rangle$ is $\{x\}^+/\rho$ where ρ is the intersection of all congruences which identify x^4 with x^2. Take the word $x^6 \in X^+$. Now we know that $(x^4, x^2) \in \rho$ therefore $x^6 = (x^4)x^2\rho(x^2)x^2 = x^4\rho x^2$ and so $(x^6, x^2) \in \rho$ that is x^6 is in the same congruence class as x^2. We would write $x^6 = x^2$.

As every object O in variety $\mathcal{V}$ is isomorphic to a quotient of a free object then there is a (non unique) presentation which will define an object which is isomorphic to O.

A semigroup presentation can be regarded as a *rewriting system*. Given a semigroup presentation $P = \langle X | U \rangle$ and given a word $w \in X^+$ with a subword u such that $(u, v) \in U$ (or $(v, u) \in U$) then we may replace the subword u with v in w to create a new word z. We would say that $w = z$. In the above example x^6 can be *rewritten* as x^2. In a *confluent* rewriting system a word w can be rewritten in its *canonical form*. The latter is some special element in the congruence class of w - usually the length-by-lexicographic least element in $w\rho$. In the above example x^6 is rewritten as x^2 which is length-by-lexicographic less than x^6.

If $\mathcal{V} \subseteq \mathcal{S}$ is a variety then $\mathbf{F}_{\mathcal{V}}(X)$ can be presented as a semigroup. In particular we have:

$$\mathbf{F}_{\mathcal{M}}(X) = \langle X \cup \{\epsilon\} | x\epsilon = \epsilon x,\ x\epsilon = x \rangle$$

and

$$\mathbf{F}_{\mathcal{G}}(X) = \langle X \cup X^{-1} \cup \{\epsilon\} | x\epsilon = \epsilon x,\ x\epsilon = x,\ xx^{-1} = x^{-1}x,\ xx^{-1} = \epsilon \rangle.$$

Of course the group (monoid) presentation of the free group (monoid) is simply $\langle X | \emptyset \rangle$.

Suppose that $P = \langle X | U \rangle$ is a $\mathcal{V}$ presentation for object O and $Q = \langle Y | V \rangle$ is the semigroup presentation for $\mathbf{F}_{\mathcal{V}}(X)$ then the semigroup presentation for O is $\langle Y | U \cup V \rangle$. In this sense monoid presentations and group presentations are shorthand for semigroup presentations.

So far we have been talking in very general terms. To be able to compute with these sort of structures we will need some finiteness conditions. Given an object O in a variety $\mathcal{V}$, we say that O is *finitely generated* if there is a presentation $\langle X | U \rangle$ of O such that X is finite. We say that O is *finitely presented* if there is a presentation $\langle X|U\rangle$ of O so that both X and U are finite. It is difficult to perform any computations with infinitely generated objects and I will not touch upon these. Likewise finite presentability is highly desirable. If in particular O is finite then O is finitely presented as we can take the generators to be the elements of O itself and the relations to be its multiplication table.

An important question is the *word problem.* This asks whether, given a certain semigroup presentation $\langle X|U\rangle$ for the semigroup S, it is generally possible to tell whether $u = v$ in S for $u, v \in X^+$. As our concern is with enumerating semigroups, we must be able to solve the word problem to be able to distinguish between elements, and so a soluble word problem is a pre-requisite for coset enumeration.

It is an interesting question whether there is always a systematic enumeration process for a semigroup presentation with solvable word problem. This question, though, is dependent on the exact meaning of "systematic". In the case of inverse semigroup presentations (see **Chapter 2** for a description of inverse semigroups and **Chapter 5** for the enumeration technique) , we must be willing to generate subsets of the inverse semigroup a number of times which is not the case for group and semigroup presentations. It is, however, clear that there must be *some* sort of enumeration process for any semigroup where the word problem is solvable as we can list the elements of X^+ in length-by-lexicographic order and work our way down them using the solution to the word problem to eliminate any words which are equal to any of the previous words. This method, though, is inferior to the coset enumeration described in the following section as clearly it is necessary to first find a method for solving the word problem, which is not always easy and is perhaps computationally inefficient.

1.3 Todd-Coxeter Coset Enumeration for Monoids

This section is based on the work but not the terminology of A. Jura [12] and B. H. Neumann [17].

The Todd-Coxeter algorithm for monoids provides us with a basic, stripped down technique. It is a useful introduction to coset enumeration although the classical algorithm was for groups.

The set of mappings of a set A into itself defines a monoid with multiplication being map composition and the identity being the identity map. We call this monoid the *transformation monoid* over A. If $A = \{1, 2, ..., n\}$ then we denote the transformation monoid over A by T_n. Similarly the set of bijections of a set A onto itself defines a group. We call this the *symmetric group* over A. If $A = \{1, 2, ..., n\}$ then we denote the symmetric group over A by S_n.

Given a monoid M, and $m \in M$, we define a map $\mu_m : M \to M$ by $\mu_m : u \mapsto um$. The map composition $\mu_m \circ \mu_n : u \mapsto umn$ and so $\mu_n \circ \mu_m = \mu_{mn}$. Therefore $T = \{\mu_m | m \in M\}$ is a monoid with $\epsilon_T = \mu_{\epsilon_M}$ and the map $\phi : M \to T$ defined by $\phi : m \mapsto \mu_m$ is an epimorphism. M can therefore be embedded in $T_{|M|}$ where $|M|$ is the number of elements in M. We call T along with mapping ϕ the *right regular representation of* M.

In exactly the same way a group G can be embedded in $S_{|G|}$ so that for $g \in G$ we define the bijection $\mu_g : G \to G$ by $\mu_g : u \mapsto ug$. In the group case all the mappings have inverses - ${\mu_g}^{-1} = \mu_{g^{-1}}$. This result was first discovered by Cayley and both the case for groups and the case for monoids are refered to as the Cayley theorem.

We now focus on monoids. If we start with a finite monoid presentation $P = \langle X | U \rangle$ for a monoid M we wish to find the following:

1. The number of elements in M.
2. The right regular representation of M acting on M.

The gist of the algorithm is that it defines *cosets* (that is cosets of the trivial submonoid $\{\epsilon\}$ of M) by post-multiplying each of the already defined cosets by each of the generators. The algorithm then "applies relations" to the cosets it has defined and identifies them with each other. The algorithm terminates if and only if M is finite.

EXAMPLE: If M is presented by $\langle x, y | \emptyset \rangle$ then the algorithm will start with the coset representing ϵ - call this coset "1". We define a new coset of 1 under the image μ_x which we shall call "2" ie. $2 := 1\mu_x$. Similarly we define $3 := 1\mu_y$. We then proceed to apply x and y to coset 2 to define cosets 4 and 5. The *coset table* will look like this:

Where $\perp$ indicates that the table is still incomplete. Clearly this procedure will not terminate as in this case M is not finite.

EXAMPLE: If M is presented by $\langle x | x^3 = x^2 \rangle$ then, as before, we start with the coset 1 representing ϵ, we then define $2 := 1\mu_x$ and $3 := 2\mu_x = 1\mu_{x^2}$ and

Cosets	x	y
1	2	3
2	4	5
3	$\perp$	$\perp$
4	$\perp$	$\perp$
5	$\perp$	$\perp$

$4 := 3\mu_x = 1\mu_{x^3}$ we then notice that we can "trace both sides of the relation $x^3 = x^2$ through our table" and we come to the conclusion that "$4 = 3$". We then "delete" coset 4 and replace all occurrences of coset 4 in the table with coset 3. The table is now complete.

Cosets	x
1	2
2	3
3	3

I shall proceed to describe this algorithm more rigorously.

1.3.1 The Data Structures

- The presentation P stored as the immutable pair of a list of generators, X and a list of pairs of words in those generators, U.
- The *set of cosets*, C which is a mutable set of positive integers. Initially $C := \{1\}$.
- The *coset table*, T which is an incomplete mutable array. The columns of T are labeled by the generators in X and the rows are labeled by C. The entries are elements of $C \cup \{\perp\}$ where $\perp$ is a symbol which tells us that the entry has yet to be considered. Entries in the table are refered to as $T(c, x)$ where $c \in C$ and $x \in X$. We define $T(\perp, x) := \perp$ for all $x \in X$. For any $w = x_1x_2...x_n \in X^*$ we recursively define $T(c, w) := T(T(c, x_1), x_2x_3...x_n)$.

- The mutable *coincidence set* $K \subseteq C \times C$. This is the set of identities of cosets which are derived from applying the relations in U.

- The *replacing function* $r : C \to C \cup \{0\}$ with $r(c) < c, \forall c \in C$.

1.3.2 The Subroutines

The full names of the subroutines are given in bold while the part of the names in italics are their shorthand names. Some procedures simply change the data structures while others return a value, others will do both.

Replace

Description: During the computation various cosets will get deleted and replaced by other cosets. Rather than physically replacing the cosets it is simpler to use a pointer (the function r) to the replacing coset (and if that coset is deleted then its pointer is used and so on). If $r(c) = 0$ then the coset c has not been deleted.

- Parameter: $c \in C$

- Locals: None

- While $r(c) > 0$ then $c := r(c)$

- Return c

Create a *New* Definition

Description: For coset c and generator x this routine defines a new coset for $T(c, x)$, modifies the data structures accordingly and returns the value of $T(c, x)$.

- Parameters: $c \in C, x \in X$.

- Local: d

Do the following in order:

- Add an element $d := \max(C) + 1$ to C.

- Add an empty row onto T labeled by d.

- Define $r(d) := 0$.
- Define $T(c, x) := d$.
- Return d.

***Identify* Coincidences**

Description: This routine works its way through the set of coincidences and modifies the data structures accordingly.

- Parameters: None.
- Locals: d_1, d_2
- While K is not empty do the following
  - Pop (c_1, c_2) from K.
  - Let $d_1 :=$ **Replace**(c_1) and let $d_2 :=$ **Replace**(c_2).
  - If $d_1 \neq d_2$ then (assuming without loss of generality that $d_1 < d_2$) do the following
    * For each entry equal to d_2 in T, replace d_2 by d_1.
    * For each $x \in X$, if $T(d_1, x) = \perp$ then replace $T(d_1, x)$ by $T(d_2, x)$ otherwise replace $T(d_1, x)$ by $\min(T(d_1, x), T(d_2, x))$ and add $(T(d_1, x), T(d_2, x))$ to K.
    * Replace all pairs (s, d_2) and (d_2, s) in K with (s, d_1) and (d_1, s) respectively.
  - Let $r(d_2) := d_1$

1.3.3 The Main Procedure

- Input: A presentation $P = \langle X | U \rangle$
- Let $c := 1$
- Repeat
  - For each $x \in X$ do **New**(c, x)

- For each $1 \leq d \leq c$ and for each $(u, v) \in U$ do the following
  - If $T(d, u) = m \neq \perp$ and $T(d, v) = n \neq \perp$ then push (m, n) onto K
- **Identify**
- Let $c := c + 1$

- Until $T(c, x) \neq \perp$ for every $c \in C$ with $r(c) = 0$ and $x \in X$
- **Tidy Up** T
- Return T

1.3.4 Comments

There are several other points to make about the enumeration algorithm.

- The **Tidy Up** procedure simply removes the rows of deleted cosets from T and renumbers C so that the rows of T are numbered $1, 2, 3...$
- The For loop over which the variable d runs is necessary because changes in the data structures made in **Identify** mean that some cosets in $\{d \in C | d < c\}$ may now trace relations.
- It is very easy to adapt this algorithm to semigroup presentations. It is only necessary to remove the first coset, which represents the identity, from the table. The monoid algorithm is more general than the semigroup algorithm in the sense that it allows presentations which involve the identity.

The proof of the following can be found in Jura [12].

Theorem 1.3.1. *The monoid coset enumeration algorithm terminates if and only the monoid M presented by P is finite in which case it returns a table with $|M|$ rows with the transformation $\mu_x : M \to M$ being represented by the x column in T.*

Briefly the proof shows that given any word in $w \in X^*$ there exists a positive integer N_w such that after a finite number of iterations, $T(1, w) = N_w$. Furthermore the first k rows of table T will stabilze in a finite number of iterations as the holes will fill up and the entries will only ever be replaced by lesser values. Therefore for each w after a finite number of iterations there will be a stable row labelled by N_w such that $T(1, w) = N_w$.

1.4 Enumeration of Right Congruence Classes

In the last section I described how to enumerate M by enumerating the cosets of the trivial submonoid $\{\epsilon\}$. In other words the monoid enumeration algorithm enumerates M/ρ where ρ is the trivial (right) congruence. The algorithm naturally extends to right congruences in general.

The new algorithm will describe the action of each generator $x \in X$ on $\{m\rho | m \in M\}$ and will do so by producing a table similar to the one in the standard algorithm.

Assume we are given a monoid M presented by $P = \langle X|U \rangle$ and a right congruence ρ on M. If $m \in M$ then we define the map $\mu_m : M/\rho \to M/\rho$ by $\mu_m : u\rho \to (um)\rho$. The composition of maps $\mu_m \circ \mu_n : u\rho \mapsto (umn)\rho$ and so $\mu_m \circ \mu_n = \mu_{mn}$.

NOTE: It should be noted that $T = \{\mu_m | m \in M\}$ and M/ρ are not "isomorphic", indeed M/ρ is not even necessarily a monoid even though T is. To see this consider the map $\theta : T \to M/\rho$ where $\theta : \mu_m \mapsto m\rho$ for $m \in M$. If $(m, n) \in \rho$ and $\mu_m \neq \mu_n$ then $\mu_m\theta = m\rho = n\rho = \mu_n\theta$ and so if this happens θ is not an injection.

The right congruence monoid algorithm starts with two inputs.

1. A finite presentation $P = \langle X|U \rangle$ for the monoid M.

2. A finite set π of pairs in $X^* \times X^*$ which generate the right congruence ρ. That is ρ is the intersection of all right congruences which contain π.

The main procedure is modified thus:

- Input: A presentation $P = \langle X|U \rangle$ and a right congruence generator π.

- Let $c := 1$
- Repeat
  - For each $x \in X$ do **New**(c, x)
  - For each $(u, v) \in \pi$ do the following
    * If $T(1, u) = m \neq \perp$ and $T(1, v) = n \neq \perp$ then Push (m, n) onto K
  - For each $1 \leq d \leq c$ and for each $(u, v) \in U$ do the following
    * If $T(d, u) = m \neq \perp$ and $T(d, v) = n \neq \perp$ then Push (m, n) onto K
  - **Identify**
  - Let $c := c + 1$
- Until $T(c, x) \neq \perp$ for every $c \in C$ and every $x \in X$.
- **Tidy Up** T
- Return T

The only change is the addition of the second For loop. In essence it is only necessary to check the application of a right congruence generator on the first coset. The reason for this is that the first coset, 1, represents the ρ-class containing the identity. If $u\rho v$ then clearly $\epsilon u \rho \epsilon v$ and indeed ϵ is the only element of M which we can *a priori* multiply on the left with. If, in the above case, $wu\rho wv$ for some $w \in X^*$ then eventually the algorithm will generate cosets representing either wu or wv and so it will discover that $T(1, wu) = T(1, wv)$.

1.5 The Todd-Coxeter Algorithm for Groups

The classical algorithm was for groups even though it is more natural in the monoid case. This is because there has been far more research done in computational group theory than there has been in computational monoid and semigroup theory for the simple reason that monoids and semigroups are much more general and don't have certain properties. For example, an important property of groups is that the order of a subgroup of a finite group G divides the order of G. This does not hold for monoids and semigroups.

The differences with the monoid algorithm (without the right congruence) are as follows:

1. The columns of the coset table are labelled by $X \cup X^{-1}$.

2. For every new definition made, $d := T(c, x)$, then we set $T(d, x^{-1}) := c$.

There are certain things to note about the group algorithm and its use:

1. It is usual to write a relation $u = v$ as the *relator* $uv^{-1} = \epsilon$, and so the presentation becomes a set of generators and a set of words. The algorithm therefore checks an individual word, u and forces $T(c, u) = c$ for each coset $c \in C$. Given any relator $u = x_1 x_2 ... x_n$ then any cyclic permutation of u is simply
$$\begin{aligned} x_i ... x_n x_1 ... x_{i-1} &= ({x_{i-1}}^{-1} ... {x_1}^{-1}) u (x_1 ... x_{i-1}) \\ &\rho ({x_{i-1}}^{-1} ... {x_1}^{-1}) \epsilon (x_1 ... x_{i-1}) \\ &= \epsilon, \end{aligned}$$
and so we may replace any relator with any of its cyclic permutations.

2. There is an algorithm for enumerating equivalence classes of a right congruence on a group which works exactly the same way as the right congruence on a monoid algorithm works.

3. If ρ is a right congruence on the group G then $\epsilon\rho$ is a subgroup of G. We may therefore think of the right congruence algorithm as enumerating the cosets of a subgroup. Hence the origins of the term *coset enumeration.*

4. The group algorithm has the same terminating conditions as the monoid algorithm. That is the algorithm terminates if and only if G (or G/ρ for the right congruence algorithm) is finite.

Chapter 2

Inverse Monoids

In this chapter I intend to get to grips with what inverse monoids are and sketch a theory of computing in inverse monoids. All proofs are taken from Petrich [18].

2.1 Green's Relations

There are several important structural properties of semigroups and monoids which are worth reminding ourselves about.

NOTATION: If S is a semigroup then S^1 is the semigroup S with an extra element, ϵ added to it which obeys the identity law G2. Clearly S^1 is always a monoid. Note that if S is already a monoid with identity η then $\epsilon\eta = \eta\epsilon = \eta$ in S^1.

Definition 2.1.1. Let S be a semigroup. If $e \in S$ and $e = e^2$ then we call e an *idempotent*. We denote the set of idempotents in S by E_S.

Definition 2.1.2. Let S be a semigroup and let $s, t \in S$. We define the following equivalence relations on S.

- $s\mathcal{R}t$ if and only if there exists $u, v \in S^1$ such that $su = t$ and $tv = s$. We write R_s for the $\mathcal{R}$-class containing s.
- $s\mathcal{L}t$ if and only if there exists $w, x \in S^1$ such that $ws = t$ and $xt = s$. We write L_s for the $\mathcal{L}$-class containing s.

- $s\mathcal{J}t$ if and only if there exists $u, v, w, x \in S^1$ such that $wsu = t$ and $xtv = s$. We write J_s for the $\mathcal{J}$-class containing s.
- $\mathcal{H} = \mathcal{L} \cap \mathcal{R}$. We write H_s for the $\mathcal{H}$-class containing s.

These equivalences are called *Green's relations.*

Lemma 2.1.3. *If S is a semigroup then $\mathcal{L}$ is a right congruence on S and $\mathcal{R}$ is a left congruence on S.*

PROOF: Suppose $s\mathcal{L}t$ in S then $s = ut$ and $t = vs$ for some $u, v \in S^1$. We therefore have $sw = (ut)w$ and $tw = (vs)w$ and therefore $sw\mathcal{L}tw$ for any $w \in S$. $\mathcal{L}$ is therefore a right congruence on S and dually $\mathcal{R}$ is a left congruence on S. ■

Lemma 2.1.4. $\mathcal{L} \circ \mathcal{R} = \mathcal{R} \circ \mathcal{L}$

PROOF: Let $s\mathcal{L}u$ and $u\mathcal{R}t$ for $s, t, u \in S$. Now $s = wu$, $t = ux$, $u = ys = tz$ for some $w, x, y, z \in S^1$. Let $v = sx = wt$, $s = wu = wtz = vz$, $t = ux = ysx = yv$ and so $s\mathcal{R}v$ and $v\mathcal{L}t$ so that $\mathcal{L} \circ \mathcal{R} \subseteq \mathcal{R} \circ \mathcal{L}$ and dually $\mathcal{L} \circ \mathcal{R} \subseteq \mathcal{R} \circ \mathcal{L}$. ■

Finally we define the Green's relation $\mathcal{D} = \mathcal{R} \circ \mathcal{L}$. For any given semigroup $\mathcal{H} \subseteq \mathcal{R} \subseteq \mathcal{D} \subseteq \mathcal{J}$. Dually $\mathcal{H} \subseteq \mathcal{L} \subseteq \mathcal{D} \subseteq \mathcal{J}$.

$$\begin{tikzcd}[arrows=-] & \mathcal{J} \arrow[d] & \\ & \mathcal{D} \arrow[dl] \arrow[dr] & \\ \mathcal{L} \arrow[dr] & & \mathcal{R} \arrow[dl] \\ & \mathcal{H} & \end{tikzcd}$$

$\mathcal{R}$-classes and $\mathcal{L}$-classes of a semigroup, S have certain desirable features when they contain idempotents. The two following somewhat technical lemmas will be important when we consider the notion of inverses in semigroups. The proofs are not difficult and I would refer the reader to a standard text on semigroup theory, for example J. M. Howie [9].

Lemma 2.1.5. *Let S be a semigroup and let $s, t \in S$. Then $st \in R_s \cap L_t$ if and only if $L_s \cap R_t$ contains an idempotent. In such a case,*

$$sH_t = H_st = H_sH_t = H_{st} = R_s \cap L_t.$$

Lemma 2.1.6. *Let $e, f \in E_S$. For every $s \in R_e \cap L_f$, there exists a unique $t \in R_f \cap L_e$ such that $st = e$ and $ts = f$.*

Corollary 2.1.7. *If S is a semigroup and $e \in E_S$, then H_e is a group.*

PROOF: By Lemma 7.1.3 we have ${H_e}^2 = H_e$ so H_e is closed under semigroup multiplication. Given $s \in H_e$ then by letting $f = e$ in Lemma 2.1.6 we know that there exists a unique $t \in H_e$ such that $st = ts = e$. Thus $se = s(ts) = (st)s = es$ and we know that because $s\mathcal{R}e$ that there exists $u \in S^1$ such that $s = eu$ and so $es = e^2u = eu = s$ and so e is an identity for H_e and t is an inverse for s. ■

It is at first quite surprising that the variety of semigroups which is so general has some definite, general structural properties. In particular it is important to understand that different $\mathcal{R}$-classes (and dually $\mathcal{L}$-classes) within a given $\mathcal{D}$-class are structurally identical. We have the following vital lemma.

Lemma 2.1.8 (Green's lemma). *Let s and t be $\mathcal{L}$-related elements of a semigroup S. By hypothesis there exist $u, u' \in S^1$ such that $us = t$ and $u't = s$. The mappings*

$$\sigma : x \mapsto ux \ (x \in R_s)$$

and

$$\sigma' : y \mapsto u'y \ (y \in R_t),$$

are mutually inverse, $\mathcal{L}$-class preserving bijections of R_s and R_t.

PROOF: If $x \in R_s$ then $ux\mathcal{R}us$. As $t = us$ so $ux \in R_t$. Hence σ maps R_s into R_t. Similarly σ' maps R_t into R_s.

For any $x \in R_s$, we have $x = sv$ for some $v \in S^1$ and thus

$$x\sigma\sigma' = u'ux = u'u(sv) = u'(us)v = u'tv = sv = x.$$

Hence $\sigma\sigma'$ is the identity mapping on R_s. Similarly $\sigma'\sigma$ is the identity mapping on R_t. If $x \in R_s$, then $x\sigma = ux$ and $x = u'(x\sigma)$ so that $x\mathcal{L}x\sigma$. Hence σ is $\mathcal{L}$-class preserving. Similarly σ' is also $\mathcal{L}$-class preserving. ■

Even at this stage it is worth noting that Green's lemma could be used to "run through" the $\mathcal{R}$-classes in any given $\mathcal{D}$-class. If we start with R_u we may define other $\mathcal{R}$-classes in D_u by $\{vR_u | v \in V\}$ where V is some sort of "canonical set" of left multipliers which includes the identity.

There is of course a dual for Green's lemma where right multipliers permute $\mathcal{L}$-classes.

2.2 Regular Semigroups and Monoids

Definition 2.2.1. Let S be a semigroup and let $s \in S$. We say that s is *regular* if there exists $a' \in S$ such that $aa'a = a$ and $a'aa' = a'$. In such a case a' is an *inverse* of a. The set of inverses for a is denoted $V(a)$. We say that S is a *regular semigroup* if every $a \in S$ is regular. If S is also a monoid then we say that S is a *regular monoid.*

Given any semigroup S then an example of a regular element is any idempotent $e \in E_S$. It is easy to see that e is its own inverse.

Lemma 2.2.2. *Let S be a semigroup and let $a \in S$ be regular with $a' \in V(a)$. Then the following hold:*

(i) $a\mathcal{R}aa'$.

(ii) $a\mathcal{L}a'a$.

(iii) $a\mathcal{D}a'$.

(iv) aa' *and* $a'a$ *are both idempotent.*

PROOF: (i) We need to find u and v in S^1 such that $au = aa'$ and $aa'v = a$. Let $u = a'$ and $v = a$.

(ii) We need to find u and v in S^1 such that $ua = a'a$ and $va'a = a$. Let $u = a'$ and $v = a$.

(iii) By (i) and (ii) and noting that a is an inverse of a', $a\mathcal{R}aa'\mathcal{L}a'$ and so $a\mathcal{D}a'$.

(iv) $(aa')(aa') = (aa'a)a' = aa'$, $(a'a)(a'a) = (a'aa')a = a'a$. ■

Theorem 2.2.3. *Let S be a semigroup, the following two statements are equivalent.*

(i) S is a regular semigroup.

(ii) Every $\mathcal{L}$-class and every $\mathcal{R}$-class in S has at least one idempotent.

PROOF: (i) $\Rightarrow$ (ii). This follows from Lemma 2.2.2 parts (i), (ii) and (iv).

(ii) $\Rightarrow$ (i). Let $a \in S$, let e be an idempotent in L_a and let f be an idempotent in R_a. Now $ua = e$, $ve = a$, $ax = f$ and $fy = a$ for some $u, v, x, y \in S^1$. I want to show that uax is an inverse for a. We have

$$a(uax)a = a(ua)xa = aexa = (ve)exa = vexa = (ax)a = fa = f(fy) = a$$

and

$$(uax)a(uax) = u(ax)(a)(ua)x = uf(fy)ex = uaex = u(ve)ex = uax$$

as required. ■

If we recall Todd-Coxeter enumeration methods for groups, the action of the group on itself on the right is examined systematically. Now there is no free regular semigroup over a set X and so there are no regular semigroup presentations, so there is special Todd-Coxeter method for regular semigroups as a class as Todd-Coxeter requires a presentation for the input. However before moving on to inverse semigroups let us have a quick look at $\mathcal{R}$-classes of regular semigroups.

Let S be a regular semigroup. Let $a \in S$ and let $a' \in V(a)$. Now $aa'\mathcal{R}a$ by Lemma 2.2.2 (i) and so $R_a = R_{aa'}$. Consider the subset of S

$$U_a = \{u | aa'\mathcal{R}aa'u, \exists u' \in V(u) \text{ such that } aa'uu' = uu'aa'\}.$$

For $u \in U_a$, $aa'uu' = uu'aa'\mathcal{L}aa'$ and so for some $v \in S^1$ where $aa'uv = aa'$ we have:

$$aa'uu' = uu'aa' = uu'aa'uv = aa'uu'uv = aa'uv = aa'.$$

Hence within R_a we have inverses working somewhat like inverses in groups as long as we only act on the right within this subset U_a of S. This is not strong enough for the systematic approach of Todd-Coxeter style algorithm for $\mathcal{R}$-classes special to regular semigroups and monoids because of the multitude of inverses any particular element has. We need a more refined class of semigroups before we can approach this question.

2.3 Introduction to Inverse Semigroups

Definition 2.3.1. A regular semigroup, S, is an *inverse semigroup* if every $a \in S$ has a unique inverse. If S is also a monoid then we call S an *inverse monoid*. The inverse of a is written a^{-1}.

NOTE: If $a \in S$ then by definition $(a^{-1})^{-1} = a$.

There are two alternative definitions of inverse semigroups (inverse monoids) equivalent to the above definition summarized in the following theorem.

Theorem 2.3.2. *Let S be a semigroup, the following statements are eqivalent:*

(i) S is an inverse semigroup.

(ii) S is regular and its idempotents commute.

(iii) Every $\mathcal{L}$-class and every $\mathcal{R}$-class of S contain exactly one idempotent.

PROOF: (i) $\Rightarrow$ (ii). Let $e, f \in E_S$ and $a = (ef)^{-1}$. Now $(ae)(ef)(ae) = (a(ef)a)e = ae$ and $(ef)(ae)(ef) = (ef)a(ef) = ef$ and so $ae = (ef)^{-1}$, likewise $fa = (ef)^{-1}$ and so $a = ae = fa$. But then $a^2 = (ae)(fa) = a(ef)a = a$ so that $ef = a^{-1} = a \in E_S$ by Lemma 2.2.1 (vi). Symmetrically $fe \in E_S$. Consequently $(ef)(fe)(ef) = efef = ef$ and $(fe)(ef)(fe) = fefe = fe$ and so $(ef)^{-1} = fe$. But we know that $(ef)^{-1} = ef$ and so $fe = ef$.

(ii) $\Rightarrow$ (iii). By way of contradiction let $e, f \in E_S$ be $\mathcal{L}$-related. Then $e = uf$ and $f = ve$ for some $u, v \in S^1$, so that $ef = fe^2 = efe = uf^2uf = (uf)^2 = e^2 = e$ and similarly $ef = ef^2 = ve^2ve = (ve)^2 = f^2 = f$ as required. Similarly $e\mathcal{R}f$ implies $e = f$.

(iii) $\Rightarrow$ (i). By Theorem 2.2.3 we know that S is regular. Let x and y be inverses of an element a of S. Then $xa, ya \in E_S$ and $xa\mathcal{L}a\mathcal{L}ya$ and thus by hypothesis, $xa = ya$. Symmetrically, we get $ax = ay$. Hence $x = xax = yay = y$ as required. ■

Corollary 2.3.3. *Let S be an inverse semigroup and let D be a $\mathcal{D}$-class of S. Then the $\mathcal{R}$-classes and $\mathcal{L}$-classes of D are in one-to-one correspondence with each other.*

We can now write a universal algebra style definition for inverse semigroups.

Definition 2.3.4. An *inverse semigroup*, S, is a semigroup with a unary operation $^{-1}$ so that for any $x, y \in S$ the following axioms hold:

(IS1) $(x^{-1})^{-1} = x$

(IS2) $(x * y)^{-1} = y^{-1} * x^{-1}$

(IS3) $x * x^{-1} * x = x$

(IS4) $x * x^{-1} * y * y^{-1} = y * y^{-1} * x * x^{-1}$

An *inverse monoid, M* is an inverse semigroup with an identity ϵ_M which satisfies G2.

By examining the latter definition it is clear that both inverse semigroups and inverse monoids form varieties.

NOTATION: The variety of inverse semigroups is denoted $\mathcal{IS}$ and the variety of inverse monoids is denoted $\mathcal{IM}$.

It is clear that all groups are inverse monoids and all inverse monoids are inverse semigroups. So we have the lattice of varieties:

$$\begin{tikzcd}[arrows=dash]
 & \mathcal{S} & \\
\mathcal{IS} \arrow[ur] & & \mathcal{M} \arrow[ul] \\
 & \mathcal{IM} \arrow[ul] \arrow[ur] \arrow[d] & \\
 & \mathcal{G} &
\end{tikzcd}$$

Using Definition 2.3.4 we can define homomorphisms between inverse semigroups and inverse monoids.

Definition 2.3.5. If S and T are two inverse semigroups, an *inverse semigroup homomorphism* is a map $\mu : S \to T$ such that for $x, y \in S$:

(1) $(xy)\mu = x\mu y\mu$

(2) $x^{-1}\mu = (x\mu)^{-1}$

If M and N are two inverse monoids then an inverse monoid homomorphism $\nu : M \to N$ will satisfy the above properties as well as

(3) $\epsilon_M \nu = \epsilon_N$.

Lemma 2.3.6. *Given two inverse semigroups (inverse monoids) S and T, and a map $\mu : S \to T$ then μ is an inverse semigroup (inverse monoid) homomorphism if and only if $(xy)\mu = x\mu y\mu, \forall x, y \in S$ (and $\epsilon_S\mu = \epsilon_T$). That is it is only necessary to check conditions (1) and (3).*

PROOF: I shall check condition (2) assuming condition (1). Let $x \in S$. Define $y = x^{-1}\mu$. Now $x\mu = (xx^{-1}x)\mu = (x\mu)y(x\mu)$ and $y = x^{-1}\mu = (x^{-1}xx^{-1})\mu = y(x\mu)y$ and so y is an inverse of $x\mu$ in T, hence by uniqueness of inverses $x^{-1}\mu = y = (x\mu)^{-1}$. ∎

Lemma 2.3.7. *Let S be an inverse semigroup then*

(i) *each idempotent in S is its own inverse,*

(ii) *for each $a \in S$, aa^{-1} is an idempotent,*

(iii) *each idempotent in S is the product of an element and its inverse.*

(iv) *the set of idempotents of S forms a semilattice, that is it is a closed algebraic structure where every element is idempotent and $ef = fe$ for $e, f \in E_S$.*

PROOF:

(i) Let $e \in E_S$. By Definition 2.3.1 we have a unique inverse e^{-1} of e satisfying $ee^{-1}e = e$ and $e^{-1}ee^{-1} = e^{-1}$, however e satisfys these conditions for e^{-1} and so $e^{-1} = e$ by uniqueness of inverses.

(ii) Given $a \in S$ then by (IS3) in Definition 2.3.4,

$$(aa^{-1})(aa^{-1}) = (aa^{-1}a)a^{-1} = aa^{-1}.$$

(iii) Let $e \in E_S$. By (i), $e^{-1} = e$ and so $ee^{-1} = e^2 = e$.

(iv) Given any $e, f \in E_S$, by (iii) e and f are the products of elments and their inverses and by (IS4) in Definition 2.3.4, $(ef)^2 = efef = e^2f^2 = ef$ and so E_S is a subsemigroup of S. Clearly each element of E_S is idempotent and again by (IS4) each pair of elements commute hence satisfying the semilattice axioms.

■

At the end of **Section 2.2** we defined a subset of a regular semigroup S as follows:

$$U_a = \{u | aa'\mathcal{R}aa'u, \exists u' \in V(u) \text{ such that } aa'uu' = uu'aa'\}.$$

If S is an inverse semigroup then U_a is more simply

$$U_a = \{u | aa^{-1}\mathcal{R}aa^{-1}u\}.$$

For inverse semigroups $aa^{-1}U_a = R_a$, whereas for regular semigroups $aa'U_a$ is only a subset of R_a, suggesting that enumerating R_a is similar to enumerating a group providing that we have a test for $\mathcal{R}$-equivalence. Indeed this conjecture is born out even further by the Wagner representation theorem which shows how inverse semigroups can be represented when they act on themselves.

2.4 Wagner Representation Theorem

Analagous to the Cayley theorem for groups and the Cayley theorem for semigroups we have the Wagner representation theorem [31] for inverse semigroups which states the intimate relation between inverse semigroups and *partial injections*. Firstly though, some definitions are needed.

Definition 2.4.1. Given a set X, a *partial transformation*, $\tau : X \to X$, is a mapping of a subset of X into X. Likewise a *partial injection*, $\iota : X \to X$, is an injection of a subset of X into X. Let α be a partial transformation on X we denote the domain of α by $\mathbf{d}(\alpha)$ and the range of α by $\mathbf{r}(\alpha)$.

Definition 2.4.2. For a set X, the *symmetric inverse monoid* over X is the set of all partial injections $\iota : X \to X$ with composition written on the right. It is denoted $\mathcal{I}(X)$. The set of all partial transformations over X is denoted $\mathcal{F}(X)$.

NOTE: The similar set where the compositions are written on the left is antiisomorphic to $\mathcal{I}(X)$. (See Lemma 2.4.3 below for the proof that $\mathcal{I}(X)$ is a monoid.)

The name symmetric inverse monoid comes from the name of the symmetric group or full permutation group . The reader is encouraged to remember that a group, G, acting on itself induces a group of permutations of G isomorphic to G

(Cayley's theorem). One sometimes refers to partial injections as *partial symmetries*.

Given a set of symbols, X, let us add another symbol 0 and call the new set X_0. Given a partial transformation, α on X we may convert this to a transformtion α' on X_0 by defining $\alpha' : X_0 \setminus \mathbf{d}(\alpha) \to \{0\}$ and $\alpha' : x \mapsto x\alpha, \forall x \in \mathbf{d}(\alpha)$. We may therefore think of partial transformations on X as tranformations on X_0 such that 0 is always mapped to 0, and throughout this thesis I shall treat them as such objects.

Lemma 2.4.3. *$\mathcal{F}(X)$ is a monoid. $\mathcal{I}(X)$ is an inverse submonoid of $\mathcal{F}(X)$.*

PROOF: Considering partial transformations on X as transformations on X_0 it is easy to see that they are well defined and that the composition is associative. To see that the composition of two partial transformations gives another partial transformation notice that all that is needed is that the compostion maps 0 to 0 and as both partial transformations map 0 to 0 then the composition certainly does. Finally note that the identity transformation is a partial transformation and we have that $\mathcal{F}(X)$ is a monoid, or to be more precise a submonoid of T_{X_0}.

Consider $\alpha \in \mathcal{I}(X)$. Now α restricted to $\mathbf{d}(\alpha)$ is a bijection from $\mathbf{d}(\alpha)$ to $\mathbf{r}(\alpha)$ and hence has an inverse, $\alpha|_{\mathbf{d}(\alpha)}{}^{-1}$, we construct an inverse, α' for α by extending $\alpha|_{\mathbf{r}(\alpha)}{}^{-1}$ to X_0 by defining $x\alpha' = 0$ for $x \in X_0 \setminus \mathbf{r}(\alpha)$. Note that $x\alpha' = 0$ if and only if $x \in X_0 \setminus \mathbf{r}(\alpha)$. To check that $\alpha\alpha'\alpha = \alpha$ we need to consider the following two cases:

- $x \in X_0 \setminus \mathbf{d}(\alpha)$ in which case it is easy to see that both sides of the equation map x to 0.
- $x \in \mathbf{d}(\alpha)$ in which case $x\alpha \neq 0$ and so $x\alpha\alpha' = x$ and so $x\alpha\alpha'\alpha = (x\alpha\alpha')\alpha = x\alpha$ as required.

To check that $\alpha'\alpha\alpha' = \alpha'$ we need to consider the following two cases:

- $x \in X_0 \setminus \mathbf{r}(\alpha)$ in which case it is easy to see that both sides of the equation map x to 0.
- $x \in \mathbf{r}(\alpha)$ in which case $x\alpha' \neq 0$ and so $x\alpha'\alpha = x$ and so $x\alpha'\alpha\alpha' = (x\alpha'\alpha)\alpha' = x\alpha'$ as required.

We now have a method for constructing an inverse for every element of $\mathcal{I}(X)$ which we may consider a unary operation (although these inverses are not necessarily unique). We may talk about α' and β' as inverses of α and β.

Observe that both $\alpha\alpha'$ and $\alpha'\alpha$ will either map an element of X_0 to itself or to 0. These maps are both identities on a subset of X and map the rest of X to 0. It is therefore clear that the product of $\alpha\alpha'$ and $\beta\beta'$ will commute. The unary operation of constructing inverses thus fulfills axioms (IS1), (IS3) and (IS4) of Definition 2.3.4 and by the note on (IS2) this is all that is required to show that $\mathcal{I}(X)$ is an inverse monoid. ■

Lemma 2.4.4. *Given a set X and $\alpha, \beta \in \mathcal{I}(X)$ then $\mathbf{d}(\alpha\beta) = (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\alpha^{-1}$ and $\mathbf{r}(\alpha\beta) = (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\beta$.*

PROOF: Let $x \in \mathbf{d}(\alpha\beta)$ then clearly $x\alpha \in \mathbf{r}(\alpha)$ and $x\alpha \in \mathbf{d}(\beta)$ hence $x \in (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\alpha^{-1}$. Conversely let $x \in (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\alpha^{-1}$ then $x\alpha \in \mathbf{r}(\alpha)$ and $x\alpha \in \mathbf{d}(\beta)$ and so $x \in \mathbf{d}(\alpha\beta)$. Hence $\mathbf{d}(\alpha\beta) = (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\alpha^{-1}$.

Similarly it follows directly from definition that $\mathbf{r}(\alpha\beta) = (\mathbf{r}(\alpha) \cap \mathbf{d}(\beta))\beta$. ■

The Wagner Representation Theorem essentially declares that the action of an inverse semigroup, S, on itself gives partial symmetries of S.

Theorem 2.4.5 (Wagner Representation Theorem). *Let S be an inverse semigroup. For each $a \in S$ then we construct the partial symmetry on S, w^a as follows:*

$$w^a : x \mapsto xa, (x \in Sa^{-1})$$

$$w^a : x \mapsto 0, (x \in S_0 \setminus Sa^{-1}).$$

The mapping

$$w : a \mapsto w^a, (a \in S)$$

is a monomorphism of S into $\mathcal{I}(S)$.

PROOF: First note that for $a \in S$, $\mathbf{d}(w^a) = Sa^{-1} = Saa^{-1}$ and $\mathbf{r}(w^a) = Sa = Sa^{-1}a$.

If $x, y \in Saa^{-1}$ with $xa = ya$ then $xaa^{-1} = yaa^{-1}$ but as $x \in Saa^{-1}$ then $x = uaa^{-1}$ for some $u \in S$ and so $xaa^{-1} = u(aa{-}1)^2 = uaa^{-1} = x$ and similarly $yaa^{-1} = y$ and so $x = y$. Hence w is a well-defined map of S into $\mathcal{I}(S)$.

Let $a, b \in S$. Consider $x \in \mathbf{d}(w^{ab}) = S(ab)(ab)^{-1} = Sabb^{-1}a^{-1}$. Then $x = xabb^{-1}a^{-1}$ and so

$$xaa^{-1} = x(abb^{-1}a^{-1})aa^{-1} = xabb^{-1}a^{-1} = x$$

and

$$xabb^{-1} = x(abb^{-1}a^{-1})abb^{-1} = xa(bb^{-1})(a^{-1}a)(bb^{-1}) = x(abb^{-1}a^{-1})a = xa$$

and so $xa \in Sabb^{-1}$ so that $x = xaa^{-1} \in Sabb^{-1}a^{-1}$ and therefore $x \in Saa^{-1} \cap Sabb^{-1} = (Sa^{-1}a \cap Sbb^{-1})w^{a^{-1}}$ and from Lemma 2.4.4 it follows that $x \in \mathbf{d}(w^a w^b)$. Conversely let $x \in \mathbf{d}(w^a w^b)$. Then by Lemma 2.4.4 $x = xaa^{-1}$ and $xa = xabb^{-1}$. Hence

$$x = xaa^{-1} = xa(bb^{-1})a^{-1} = x(ab)(ab)^{-1}$$

and thus $x \in \mathbf{d}(w^{ab})$. We have that $\mathbf{d}(w^a w^b) = \mathbf{d}(w^{ab})$ and it is clear that if $x \in \mathbf{d}(w^a w^b) = \mathbf{d}(w^{ab})$ then $xw^a w^b = xab = xw^{ab}$ and if $x \notin \mathbf{d}(w^a w^b) = \mathbf{d}(w^{ab})$ then $xw^a w^b = 0 = xw^{ab}$. Hence $w^a w^b = w^{ab}$ and w is an inverse semigroup homomorphism.

Assume that $w^a = w^b$ for some $a, b \in S$. Then $Saa^{-1} = Sbb^{-1}$ and so

$$(aa^{-1})(bb^{-1}) = (aa^{-1})(aa^{-1}) = aa^{-1}$$

and

$$(aa^{-1})(bb^{-1}) = (bb^{-1})(aa^{-1}) = (bb^{-1})(bb^{-1}) = bb^{-1}$$

that is $aa^{-1} = bb^{-1}$. Since $aa^{-1} \in Sa^{-1}$, it follows that $aa^{-1}a = aa^{-1}b$, which implies $a = aa^{-1}b = bb^{-1}b = b$. Hence w is one-to-one and so is a monomorphism of S into $\mathcal{I}(S)$. ■

2.5 Inverse Monoid Presentations

Having discussed inverse semigroups as partial symmetries, we shall now begin to look at the theory of presentations for these objects.

Given a set of symbols, X, there is a *free inverse semigroup* over X written $\mathbf{F}_{\mathcal{IS}}(X)$ and a *free inverse monoid* over X written $\mathbf{F}_{\mathcal{IM}}(X)$.

Given that $^{-1}$ is an involution then $\mathbf{F}_{\mathcal{IS}}(X)$ is presented as a semigroup by

$$\langle X \cup X^{-1} | uu^{-1}u = u,\ uu^{-1}vv^{-1} = vv^{-1}uu^{-1}\ \forall u, v \in (X \cup X^{-1})^{+} \rangle$$

Similarly $\mathbf{F}_{\mathcal{IM}}(X)$ is presented as a monoid by the same presentation.

The congruence generated by these relations is called the *Wagner congruence* and is denoted by ρ_X or more simply just ρ when there is no confusion.

Now unfortunately this is an infinite presentation as there are an infinite number of elements in $(X \cup X^{-1})^{*}$. Worse still, this is the best we can possibly do. See Petrich [18] for a proof of the fact that $\mathbf{F}_{\mathcal{IS}}(X)$ cannot be finitely presented. At first sight this is disaterous, because Todd-Coxeter is applicable only to finite presentations. I shall show how this problem is overcome in **Chapter 3** and **Chapter 4**.

Definition 2.5.1. An *inverse semigroup presentation* is a presentation $\langle X | U \rangle$, where $U \subseteq (X \cup X^{-1})^{+} \times (X \cup X^{-1})^{+}$. If τ is the congruence generated by $\rho \cup U$ then the inverse semigroup correponding to the inverse semigroup presentation $\langle X | U \rangle$ is $(X \cup X^{-1})^{+}/\tau$.

Similarly:

Definition 2.5.2. An *inverse monoid presentation* is a presentation $\langle X | U \rangle$, where $U \subseteq (X \cup X^{-1})^{*} \times (X \cup X^{-1})^{*}$. If τ is the congruence generated by $\rho \cup U$ then the inverse monoid correponding to the inverse monoid presentation $\langle X | U \rangle$ is $(X \cup X^{-1})^{*}/\tau$.

As with group and monoid presentations we may think of invserse semigroup and inverse monoid presentations as being shorthand for a semigroup presentation which includes ρ in its relations.

Definition 2.5.3. Given two inverse semigroups S and T with presentations $\langle X | U \rangle$ and $\langle Y | V \rangle$ so that $(X \cup X^{-1}) \cap (Y \cup Y^{-1}) = \emptyset$ then the *inverse semigroup free product* is the inverse semigroup $S * T$ which is presented by $\langle X \cup Y | U \cup V \rangle$.

The *free inverse monoid product* of two inverse monoids presented as inverse monoids is defined in the same way.

Chapter 3

Problems With Enumerating Inverse Monoids

As I have commented, the results of the previous chapter are largely negative as far as computation is concerned. For a Todd-Coxeter style enumeration for some object O in variety $\mathcal{V}$ a presentation of O is required. This means that a thorough understanding of free objects in $\mathcal{V}$ is needed. As we have seen in **Chapter 1**, semigroups and monoids have very simple free objects where there is only one representation of any particular element in terms of the free generators. For groups the free object is only slightly more complicated. The normal form for any particular element is found by free cancellation and in Todd-Coxeter this cancellation is implicit in the computation and actually makes the process easier. For inverse semigroups and inverse monoids there is a normal form for any word in a free object but this is not trivial to find.

NOTATION: We will want to be talking about varieties with an associative binary operation such that the free object has a unique normal form for every element. I will call these varieties *UNF-varieties.*

3.1 An Approach to Enumerating Free Objects

In this section I shall introduce a simple, original algorithm to demonstrate "enumeration by identities" rather than "enumeration by relations" for the purpose of gaining insights into inverse monoid enumeration, which I will attempt in **Section 3.4** and again more thoroughly in **Chapter 5**.

In general Todd-Coxeter style coset enumeration has three subroutines:

(1) Making a new definition.

(2) Checking a relation or a right congruence.

(3) Processing coincidences.

A free inverse monoid $\mathbf{F}_{\mathcal{IM}}(X)$ is a monoid with an infinite number of relations and so process (2) has to be applied an infinite number of times. However, remembering the universal algebra definition of inverse monoids, the free inverse monoid obeys only a handful of *identities*, that is equations which hold true throughout the variety. So instead of looking at the infinite relations, we might look at identities which a certain word $w \in (X \cup X^{-1})^*$ must satisfy.

Supposing that variety $\mathcal{V}$ must satisfy the identity $p(x_1, x_2, ..., x_m) = q(y_1, y_2, ..., y_n)$, then for any object $O \in \mathcal{V}$ generated by the set X, O will satisfy the set of relations

$$\{p(u_1, u_2, ..., u_m) = q(v_1, v_2, ..., v_n) | u_1, u_2, ..., u_m, v_1, v_2, ..., v_n \in \mathbf{F}_{\mathcal{V}}(X)\},$$

which is finite only when $\mathbf{F}_{\mathcal{V}}(X)$ is finite.

For example any $G \in \mathcal{G}$ satisfies the identity $xx^{-1} = \epsilon$ which is to say that for every element $x \in G$ then the relation $xx^{-1} = \epsilon$ is implicitly satisfied. Let us call these relations which are implicit in identities *implicit relations* and any other relations *explicit relations*. For a free object of variety $\mathcal{V}$ generated by X all relations are implicit over the set of free generators.

How is it that free groups can be finitely presented as monoids if there are an infinite number of implicit relations from the identities? The simple answer is that only a finite number of relations are necessary. The "crude" monoid presentation of $\mathbf{F}_{\mathcal{G}}(X)$ is:

$$\langle X \cup X^{-1} | (\forall w \in (X \cup X^{-1})^*) \; ww^{-1} = \epsilon, \; w^{-1}w = \epsilon \rangle$$

however the *standard presentation* is

$$\langle X \cup X^{-1} | (\forall x \in X \cup X^{-1}) \; xx^{-1} = \epsilon \rangle,$$

which is clearly finite when X is finite. To see that this is a presentation of $\mathbf{F}_{\mathcal{G}}(X)$ consider any word $w = x_1x_2...x_n \in (X \cup X^{-1})^*$ then

$$ww^{-1} = x_1x_2...x_n{x_n}^{-1}{x_{n-1}}^{-1}...{x_1}^{-1}$$

and it is easy to see that this will cancel down to ϵ by repeated application of the relations in the standard presentation.

For inverse monoids the implicit relations cannot be reduced to a finite number of monoid relations (for a proof of this see Petrich [18]). This does not necessarily make our task impossible, as given any word $w \in (X \cup X^{-1})^*$ there are only a finite number of implicit relations whose left or right side are subwords of w.

There are two important points about free groups. One is that they can be finitely presented, the other is that words in free groups have a *unique normal form* which is very easily found. This means that given two words $u, v \in (X \cup X^{-1})^*$ it is possible, indeed very easy, to tell whether $u = v$ in $\mathbf{F}_{\mathcal{G}}(X)$. That is to say that the *word problem* is soluble.

Let us look at a variety where a normal form is easily found (and so the word problem is solvable) in the free objects. My example is that of semilattices with an identity, which form a variety $\mathcal{SL}^1 \subset \mathcal{IM}$ with a binary operation $\wedge$ and a nullary operation ϵ satisfying the following identities

(SL1) associativity $(x \wedge y) \wedge z = x \wedge (y \wedge z)$

(SL2) idempotency $x \wedge x = x$

(SL3) commutativity $x \wedge y = y \wedge x$

(SL4) identity $\epsilon \wedge x = x \wedge \epsilon = x$

Let X be a finite set and $\mathbf{F}_{\mathcal{SL}^1}(X)$ be the free semilattice with identity over X. Let there be a total order $\leq$ on X. Given a word $w = x_1 \wedge x_2 \wedge ... \wedge x_n \in X^*$, let $Y = \{x_1, x_2, ..., x_n\} \subseteq X$ then a unique normal form for w is the product of all the elements of Y ordered in ascending order by $\leq$. It is not hard to see how w can be manipulated using (SL2) and (SL3) to do this and that this new word is unique. Let $U(w)$ be this unique normal form of w.

An algorithm for enumerating $\mathbf{F}_{\mathcal{SL}^1}(X)$ follows.

Define the following data structures:

- The immutable *set of generators*, X.
- The *coset set* C which is a mutable set of positive integers. At the start of the algorithm $C := \{1\}$.
- The mutable *coset table* T which is an array of elements of $C \cup \{\perp\}$ (where $\perp$ is the *empty symbol*) with rows labelled by elements of C and columns labelled by elements of X. Entries in T are referred to by $T(c, x)$ where $c \in C$ and $x \in X$. Initially T is a single row of empty symbols.
- The mutable *representative set* $\Gamma \subseteq X^*$. Initially $\Gamma := \{\epsilon\}$.
- A surjection $\psi : C \to \Gamma$ with $1\psi = \epsilon$.
- A mutable coincidence set $K \subseteq C \times C$.
- The *replacing function* $r : C \to C \cup \{0\}$ with $r(c) < c$, $\forall c \in C$.

Define the following subroutines:

Replace

- Parameter: $c \in C$
- Locals: None
- While $r(c) > 0$ then $c := r(c)$
- Return c

Create a *New* definition

- Parameters: $c \in C$ and $x \in X$.
- Local: d
- Add $d := \max(C) + 1$ to C.
- Add $U((c\psi) \wedge x)$ to Γ.
- Add an empty row onto T labelled by d.

- Define $d\psi = U((c\psi) \wedge x)$
- Define $T(c, x) := d$

Check **a Coset.**

- Parameter: $c \in C$
- Local : d
- For each $d \in C$ if $c\psi = d\psi$ then add (c, d) to K.

Identify **Coincidences**

- Parameters: None
- Locals: c_1, c_2, d_1, d_2, s, x
- While K is not empty do the following
  - Pop (c_1, c_2) from K.
  - Let $d_1 =$ **Replace**(c_1) And Let $d_2 =$ **Replace**(c_2)
  - If $d_1 \neq d_2$ then (assuming without loss of generality that $d_1 < d_2$) do the following
    * For each entry equal to d_2 in T, replace d_2 by d_1.
    * For each $x \in X$, if $T(d_1, x) = \perp$ then replace $T(d_1, x)$ by $T(d_2, x)$ otherwise replace $T(d_1, x)$ by $\min(T(d_1, x), T(d_2, x))$ and add $((d_1, x), T(d_2, x))$ to K.
    * For each pair (s, d_2) or (d_2, s) in K, replace with (s, d_1) or (d_1, s) respectively.
  - Let $r(d_2) := d_1$

The main algorithm proceeds as follows:

- Repeat
- For $c \in C$ and each $x \in X$ do

- If $T(c, x) = \perp$ Then
- **New**(c, x)
- **Check**(c)
- **Identify**

- Until $\forall c \in C$ and $\forall x \in X, T(c, x) \neq \perp$

This algorithm will enumerate free semilattices with identities and it is easy to see that it applies just as easily to any free object of a UNF-variety although it will only terminate when the free object is finite.

It should be noted that this algorithm applies only to free objects. A procedure for enumerating general semilattices with identity is a simple generalisation as every finitely generated semilattice with identity is finite and hence has a unique normal form. Having said that, I have no general approach to enumerating any object of any UNF-variety.

The point is, though, that having a systematic method for enumerating free objects is a step towards enumerating general objects of that variety. Certainly if one has no method for enumerating the free object then there is no hope of anything approaching a Todd-Coxeter style enumeration for quotients of the free object.

As we noted there is a unique normal form for inverse semigroups. For our purposes it suffices to be able to find a unique representation of some kind.

3.2 Word Trees

In this section I will be working from W.D. Munn's paper *Free Inverse Semigroups* [16]. This is a graph theoretic approach to handling $\mathbf{F}_{\mathcal{IM}}(X)$ which shows how to solve the word problem.

As I will be talking consistently about $\mathbf{F}_{\mathcal{IM}}(X)$ in this section, I will refer to the Wagner congruence on X^* by ρ.

Definition 3.2.1. A *tree* is a connected, directed graph without cycles except that for every edge (α, β) there is an edge (β, α) in the opposite direction.

Usually a tree is defined to be a connected, non-directed graph without cycles. The only difference with this and the above definition is that the latter allows the two directions of an edge to be distinguished.

For any set X called a *labelling set* there is a corresponding set X^{-1} with $|X| = |X^{-1}|$ and a bijection ${}^{-1} : X \to X^{-1}$ with the image of x being x^{-1}.

Definition 3.2.2. A *word tree* T on a labelling set X is a tree, with at least one edge, satisfying the following two conditions:

(WT1) Each edge is oriented and labelled by an element of X. For every edge from α to β labelled by x then there is another edge from β to α labelled by x^{-1}. The former is refered to as (α, x, β) and the latter as (β, x^{-1}, α).

(WT2) T is *deterministic* in that for every vertex γ all edges from γ are labelled by different elements of X. Dually T is *injective* in that for every vertex γ all edges to γ are labelled by different elements of X.

The set of vertices of a word tree, T, is denoted by $V(T)$ while the set of edges is denoted by $E(T)$. A word tree is said to be *finite* if both $V(T)$ and $E(T)$ are finite.

An example of an infinite word tree is the Cayley graph of $\mathbf{F}_{\mathcal{G}}(X)$.

Definition 3.2.3. Let T be a word tree on X and let $\alpha, \beta \in V(T)$.

- An (α, β)-*walk* on T is a sequence $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ of vertices of T such that γ_{i-1} and γ_i are adjacent vertices for $i = 1, ..., n$.
- An (α, β)-walk $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ on T is said to *span* T or to be a *spanning* (α, β)-*walk* on T, if and only if each vertex of T occurs at least once among the γ_i.
- The (α, β)-*path* on T is the unique (α, β)-walk $(\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ on T such that no vertex of T occurs more than once among the γ_i. We denote it by $\Pi(\alpha, \beta)$. The integer n is called the *length* of $\Pi(\alpha, \beta)$.

Definition 3.2.4. Let T and T' be word trees on X. A *word tree homomorphism* $\theta : T \to T'$ is a map from $V(T)$ to $V(T')$ which preserves adjacency, orientation and labelling of edges, that is if $(\alpha, x, \beta) \in E(T)$, then $(\alpha\theta, x, \beta\theta) \in E(T')$.

Similarly a *word tree monomorphism* is a word tree homomorphism which is injective. A *word tree isomorphism* is a word tree homomorphism which is bijective. A *word tree automorphism* is a word tree isomorphism which maps a word tree T onto itself.

It turns out that word tree homomorphisms and word tree monomorphism between finite word trees are actually the same thing as any map which preserves adjacency, labelling and orientation of edges will either be one-to-one or will create a cycle.

Lemma 3.2.5 (Munn). *Let T and T' be word trees on X and let $\theta : T \to T'$ and $\phi : T \to T'$ be monomorphisms such that $\alpha\theta = \alpha\phi$ for some $\alpha \in V(T)$. Then $\theta = \phi$.*

PROOF: Choose a spanning (α, β)-walk $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ on T for any vertex β of T. By hypothesis $\gamma_0\theta = \gamma_0\phi$. Suppose that $\gamma_{i-1}\theta = \gamma_{i-1}\phi$. Then if x is the label on $\gamma_{i-1}\gamma_i$ it is also the label on both $(\gamma_{i-1}\theta)(\gamma_i\theta)$ and $(\gamma_{i-1}\phi)(\gamma_i\phi)$. Hence, by (WT2), $\gamma_i\theta = \gamma_i\phi$. Thus, by induction on i, $\gamma_i\theta = \gamma_i\phi$ for $i = 0, 1, ..., n$. Every vertex of T occurs among the γ_i and so $\phi = \theta$. ■

This last result and a bit of graph theory provide us with the following theorem.

Theorem 3.2.6 (Munn). *The only automorphism of a finite word tree T on X is the identity automorphism.*

Let $P = (\alpha = \gamma_0, \gamma_1 ... \gamma_m = \beta)$ and $Q = (\beta = \delta_0, \delta_1, ... \delta_n = \gamma)$ be , respectively, an (α, β)-walk and a (β, γ)-walk on a word tree T on X. Then we define an (α, γ)-walk PQ on T by

$$PQ = (\alpha = \gamma_0, ..., \gamma_{m-1}, \beta, \delta_1, ..., \delta_n = \gamma).$$

We now have a multiplicative operation for walks given that the former ends where the latter begins (simply by concatenating them). It is clear that this operation is associative. Also if P is an (α, α)-walk then $P^r (r \in \mathbb{N})$ is the product of r copies of P with P^0 being simply the null walk $\Pi(\alpha, \alpha)$. We also define the *inverse* P^{-1} of an (α, β)-walk $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ to be the (β, α)-walk $(\beta = \gamma_n, \gamma_{n-1}, ..., \gamma_0 = \alpha)$.

For a non-null (α, β)-walk $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$ on T, we define the element $w(P) \in (X \cup X^{-1})^*$ by

$$w(P) = x_1 x_2 ... x_n,$$

where $x_i \in (X \cup X^{-1})$ is the label of the edge $\gamma_{i-1}\gamma_i$ $(i = 1, ..., n)$. We also define $w(\Pi(\alpha, \alpha)) = \epsilon$.

We have the following results:

Lemma 3.2.7 (Munn). *Let P be an (α, β)-walk and Q a (β, γ)-walk on a word tree T on X. Then $w(PQ) = w(P)w(Q)$. Also $Q = P^{-1}$ if and only if $w(Q) = w(P)^{-1}$.*

PROOF: It is clear that $w(PQ) = w(P)w(Q)$ and that if $Q = P^{-1}$ then $w(Q) = w(P)^{-1}$.

Let $w(Q) = w(P)^{-1} = x_1x_2...x_n$ $(x_i \in X \cup X^{-1})$, and $P = (\alpha = \gamma_0, \gamma_1, ..., \gamma_n = \beta)$, $Q = (\beta = \delta_0, \delta_1, ..., \delta_n = \gamma)$. We have $\delta_0 = \gamma_n$. Suppose that we have shown that $\delta_{i-1} = \gamma_{n-i+1}$. Then x_i is the label on both $\delta_{i-1}\delta_i$ and $\gamma_{n-i+1}\gamma_{n-i}$. Hence, by (WT2), $\delta_i = \gamma_{n-i}$. It follows by induction on i that $\delta_i = \gamma_{n-i}$ for $i = 0, 1, ..., n$. Hence $Q = P^{-1}$ as required. ■

It is necessary to explain exactly how word trees relate to free inverse monoids. The next lemma is technical but demonstrates that the two mathematical constructs are intimately related so I shall include Munn's proof.

Lemma 3.2.8 (Munn). *Let P and Q be spanning (α, β)-walks on a word tree T on X. Then $w(P)\rho = w(Q)\rho$.*

PROOF: First consider the case when $|T| = 2$. Let γ denote the vertex of T other than α, let $\Theta = \Pi(\alpha, \gamma)$ and let $x = w(\Theta)$ $(\in X \cup X^{-1})$. Consider the two cases

- $\beta = \alpha$. In this case $P = (\Theta\Theta^{-1})^r$, $Q = (\Theta\Theta^{-1})^s$ and so by Lemma 3.2.7,
$$w(P)\rho = (xx^{-1})^r\rho = (xx^{-1})\rho = (xx^{-1})^s\rho = w(Q)\rho.$$
- $\beta = \gamma$. In this case $P = (\Theta\Theta^{-1})^r\Theta$, $Q = (\Theta\Theta^{-1})^s\Theta$ and so by Lemma 3.2.7,
$$w(P)\rho = ((xx^{-1})^r x)\rho = x\rho = ((xx^{-1})^s x)\rho = w(Q)\rho.$$

Hence the result holds for $|T| = 2$.

Let n be a positive integer greater than 2. We make the inductive hypothesis that if P', Q' are spanning (α, β)-walks on any word tree T' on X such that $|T'| < n$ then $w(P')\rho = w(Q')\rho$.

Consider a word tree T on X such that $|T| = n$.

* Suppose that P_0 is a (γ, γ)-walk on a subtree T' of T such that $|T'| < n$, then $w(P_0)^2\rho = w({P_0}^2)\rho = (w(P_0))^2\rho$.

  To see this, let T'_0 denote the subtree of T' spanned by P_0. Then $|T'_0| < n$ and P_0 and ${P_0}^2$ are spanning (γ, γ)-walks on T'_0. Hence $w(P_0)\rho = w(P_0^2)\rho = (w(P_0))^2\rho$.

Now let P and Q be spanning (α, β)-walks on T. We have two cases.

1. α is an end point of T, that is there is only one vertex adjacent to α.

   Let γ denote the unique vertex of T adjacent to α and let T' denote the subtree of T obtained by deleting α and the edge $\alpha\gamma$ from T. We now look at the following two cases:

   - $\beta = \alpha$. Then for some (γ, γ)-walks $P_1, P_2, ..., P_k$ on T' and some non-negative integers r_i $(i = 0, ..., k)$,

     $$P = \Theta(\Theta^{-1}\Theta)^{r_0}P_1(\Theta^{-1}\Theta)^{r_1}P_2...P_k(\Theta^{-1}\Theta)^{r_k}\Theta^{-1},$$

     where $\Theta = \Pi(\alpha, \gamma)$, and so by Lemma 3.2.7 we have that

     $$w(P) = x(x^{-1}x)^{r_0}u_1(x^{-1}x)^{r_1}u_2...u_k(x^{-1}x)^{r_k}x^{-1},$$

     where $x = w(\Theta)$ and $u_i = w(P_i)$. By * we know that $u_i^2\rho = u_i\rho$. Reminding ourselves that in $\mathbf{F}_{\mathcal{IM}}(X)$, idempotents commute, we have:

     $$w(P)\rho = (x(x^{-1}x)^{r_0+r_1+...+r_k}u_1u_2...u_kx^{-1})\rho = (xu'x^{-1})\rho,$$

     where $u' = u_1u_2...u_k$. Now $u' = w(P')$ where $P' = P_1P_2...P_k$. Moreover since P spans T, it follows that P' is a spanning (γ, γ)-walk on T'.
     Similarly $w(Q)\rho = (xv'x^{-1})\rho$ where $v' = w(Q')$ for some spanning (γ, γ)-walk Q' on T'. But $|T'| = n - 1$ and so, by the inductive hypothesis, $u'\rho = v'\rho$. Hence we have

     $$w(P)\rho = (xu'x^{-1})\rho = (xv'x^{-1})\rho = w(Q)\rho$$

     as required.

- $\beta \neq \alpha$. Then $\beta \in T'$ and an argument parallel to that above shows that

  $$w(P)\rho = (xu')\rho, \;\; w(Q)\rho = (xv')\rho,$$

  where x is as before and $u' = w(P')$, $v' = w(Q')$ for some spanning (γ, β)-walks P', Q' on T'. By the inductive hypothesis, $u'\rho = v'\rho$ and so $w(P)\rho = w(Q)\rho$ as required.

2. α is not an end point of T.

   In this case we can split T into two subtrees $T_1{}'$ and $T_2{}'$ so that both trees have less than n vertices and the only common vertex is α. Again we look at two distinct cases:

   - $\beta = \alpha$. Then we write

     $$P = P_1P_2...P_r,$$

     where $P_1, P_3, P_5, ...$ are (α, α)-walks (possibly null-walks) on one of the subtrees $T_i{}'$ $(i = 1, 2)$ and $P_2, P_4, P_6, ...$ are (α, α)-walks (possibly null-walks) on the other tree. Let $u_i = w(P_i)$, $(i = 1, 2, ..., r)$. By $*$ we know that $u_i{}^2\rho = u_i\rho$, and so $w(P)\rho = (u_1{}'u_2{}')\rho$ where $u_k{}' = w(P_k{}')$ and $P_k{}'$ is the product of all the (α, α)-walks P_i on $T_k{}'$ $(k = 1, 2)$. Moreover, since P spans T, it follows that $P_k{}'$ spans $T_k{}'$ $(k = 1, 2)$.
     Similarly, we can show that $w(Q)\rho = (v_1{}'v_2{}')\rho$, where $v_k{}' = w(Q_k)$ for some spanning (α, α)-walk $Q_k{}'$ on $T_k{}'$ $(k = 1, 2)$. By the inductive hypothesis, $w(P_k{}')\rho = w(Q_k{}')\rho$ $(k = 1, 2)$.

   - $\beta \neq \alpha$. Without loss of generality, we can assume that $\beta \in V(T_2)$. By an argument similar to that above we can show that $w(P)\rho = (u_1{}'u_2{}')\rho$, $w(Q)\rho = (v_1{}'v_2{}')\rho$, where $u_1{}' = w(P_1{}')$, $v_1{}' = w(Q_1{}')$ for some spanning (α, α)-walks $P_1{}'$, $P_1{}'$ on $T_1{}'$ and $u_2{}' = w(P_1{}')$, $v_2{}' = w(Q_2{}')$ for some spanning (α, β)-walks $P_2{}'$, $Q_2{}'$ on $T_2{}'$. Hence $w(P)\rho = w(Q)\rho$.

■

The proof of the next lemma demonstrates how to construct word trees so that a spanning walk traces a particular word.

Lemma 3.2.9 (Munn). *Let $u \in \mathbf{F}_{\mathcal{IM}}(X)$. Then there exists a word tree T on X and a spanning (α, β)-walk P on T such that $u = w(P)$.*

PROOF: Let $u = x_1 x_2 ... x_n$, where $x_i \in X \cup X^{-1}$. We construct a sequence of word trees $T_1 \subseteq T_2 \subseteq ... \subseteq T_n$ on X and a sequence of vertices $\gamma_0, \gamma_1, ..., \gamma_n$ of T_n such that T_i is spanned by $P_i = (\gamma_0, \gamma_1, ..., \gamma_i)$ and $x_1 x_2 ... x_i = w(P_i)$ $(i = 1, ..., n)$.

First let T_1 denote the word tree with two vertices γ_0 and γ_1 in which $\gamma_0\gamma_1$ is labelled x_1. Now suppose that we have constructed the sequences as far as T_i and γ_i. Consider the $(i+1)$th step. There are two possibilities.

1. There exists a vertex δ in T_i, adjacent to γ_i and such that $\gamma_i\delta$ has label x_{i+1}. Then we take $T_{i+1} = T_i$ and $\gamma_{i+1} = \delta$.
2. There exists no such vertex δ in T_i with the property stated in 1. In this case we adjoin a new vertex γ_{i+1} to T_i and a new edge $\gamma_i\gamma_{i+1}$ which we label x_{i+1}. Let T_{i+1} denote the word tree so formed.

In either case T_{i+1} is spanned by $P_{i+1} = (\gamma_0, \gamma_1, ..., \gamma_{i+1})$ and $x_1 x_2 ... x_{i+1} = w(P_{i+1})$. By induction on i, the sequences can be constructed as far as T_n and γ_n. The result follows by taking $T = T_n$, $P = (\gamma_0, \gamma_1, ..., \gamma_n)$, and $\alpha = \gamma_0$, $\beta = \gamma_n$. ■

We need another couple of technical lemmas concerning isomorphisms between word trees which I shall omit the proofs of as we shall return to this topic in **Chapter 4.**

Lemma 3.2.10 (Munn). *Let T, T' be word trees on X. Let P be a spanning (α, β)-walk on T and P' an (α', β')-walk on T' such that $w(P) = w(P')$. Then there exists a monomorphism $\theta : T \to T'$ such that $\alpha\theta = \alpha', \beta\theta = \beta'$. If, in addition, P' spans T' then θ is an isomorphism.*

Lemma 3.2.11 (Munn). *Let T and T' be word trees on X. Let P be spanning (α, β)-walk on T and P' a spanning (α', β')-walk on T' such that $w(P)\rho = w(P')\rho$. Then there exist an isomorphism $\theta : T \to T'$ such that $\alpha\theta = \alpha'$, $\beta\theta = \beta'$.*

NOTATION: Let $\mathcal{T}_X$ denote a transversal of the isomorphism classes of word trees on X. Let $\mathcal{BT}_X$ denote the class of ordered triples (α, T, β), where $T \in \mathcal{T}_X$ and $\alpha, \beta \in V(T)$. We refer to any such triple as a *birooted word tree* on X.

Note that birooted word trees are deterministic inverse automata (see **Section 4.1**). It should also be remembered that these are always subsets of the Cayley graph for the free group over X with the *initial state* as the group identity and the *terminal state* as a particular word in the free group.

Theorem 3.2.12 (Munn). *If P and Q are spanning (α, β)-walks on a word tree T on X then $w(P)\rho = w(Q)\rho$ and the mapping $\phi : \mathcal{BT}_X \to \mathbf{F}_{\mathcal{IM}}(X)$ defined by*

$$(\alpha, T, \beta)\phi = w(P')\rho,$$

where P' is any spanning (α, β)-walk on T, is a bijection.

Furthermore, $((\alpha, T, \beta)\phi)^{-1} = (\beta, T, \alpha)\phi$ and $(\alpha, T, \beta)\phi$ is an idempotent if and only if $\alpha = \beta$.

PROOF: By Lemma 3.2.8, if P and Q are spanning (α, β)-walks on a word tree T on X then $w(P)\rho = w(Q)\rho$. Hence ϕ is well defined.

By Lemma 3.2.9, ϕ is surjective. To show ϕ is injective, suppose that $(\alpha, T, \beta)\phi = (\alpha', T', \beta')\phi$. Then by Lemma 3.2.11, there exists an isomorphism $\phi : T \to T'$ such that $\alpha\theta = \alpha'$, $\beta\theta = \beta'$. Thus $T = T'$, by the definition of $\mathcal{T}_X$. But now θ is an automorphism of T and so, by Theorem 3.2.6, $\alpha = \alpha'$ and $\beta = \beta'$, as required. Thus ϕ is a bijection.

Let $(\alpha, T, \beta) \in \mathcal{BT}_X$ and let P be a spanning (α, β)-walk on T. Then P^{-1} is a spanning (β, α)-walk on T and hence, by Lemma 3.2.7,

$$(\beta, T, \alpha)\phi = w(P^{-1})\rho = w(P)^{-1}\rho = ((\alpha, T, \beta)\phi)^{-1}.$$

Suppose that $(\alpha, T, \beta)\phi$ is an idempotent. Then $((\alpha, T, \beta)\phi)^{-1} = (\alpha, T, \beta)\phi$ and so, by the previous result, $(\beta, T, \alpha)\phi = (\alpha, T, \beta)\phi$. Thus $\alpha = \beta$.

Conversely, if Q is a spanning (α, α)-walk on T then so also is QQ^{-1} and therefore

$$(\alpha, T, \alpha)\phi = w(QQ^{-1})\rho = (w(Q)w(Q)^{-1})\rho,$$

which is an idempotent. ■

With Theorem 3.2.12 we can now talk about the unique up to isomorphism birooted word tree corresponding to the word $u \in \mathbf{F}_{\mathcal{IM}}(X)$. We denote this birooted word tree as the triple (α_u, T_u, β_u).

The next lemma is proved in a more general fashion in Corollary 4.2.8.

Lemma 3.2.13 (Munn). *Let $u = x_1x_2...x_n$ and let $u' = x_1x_2...x_m$ where $m \leq n$, $(x_i \in X \cup X^{-1})$. Let (α_u, T_u, β_u) and $(\alpha_{u'}, T_{u'}, \beta_{u'})$ be birooted word trees corresponding to u and u' respectively. Then there is a monomorphism $\theta : T_{u'} \to T_u$.*

Conversely, let (α, T, β) and (α', T', β') be two birooted word trees such that there is a monomorphism $\theta : T' \to T$ so that $\alpha'\theta = \alpha$. Then there is a spanning (α, β)-walk P on T so that given any spanning (α', β')-walk P' on T', $w(P)\rho = (x_1x_2...x_n)\rho$ and $w(P')\rho = (x_1x_2...x_m)\rho$ for $m \leq n$ $(x_i \in X \cup X^{-1})$.

EXAMPLE: Let $X = \{x, y\}$ and let $u = x^2x^{-1}yx$. Then the birooted word tree (α_u, T_u, β_u) is:

$$\begin{tikzcd} & & \gamma_3 \arrow[r, "x"] & \beta_u \arrow[r] & {} \\ {} \arrow[r] & \alpha_u \arrow[r, "x"] & \gamma_1 \arrow[u, "y"] \arrow[r, "x"] & \gamma_2 & \end{tikzcd}$$

I have, of course, left out the "inverse edges" (for example $(\gamma_2, x^{-1}, \gamma_1)$) as it is quite natural to read these as the other edges going backwards. To emphasise α_u as the "input" vertex and β_u as the "output" vertex, I have used the standard automata notation of putting an extra arrow pointing to the input and an extra arrow pointing from the output.

EXAMPLE: If $X = \{x, y\}$ and $u = yy^{-1}x^2x^{-2}$ then the birooted word tree $(\alpha_u, T_u, \alpha_u)$ (without labelled vertices) is:

$$\begin{tikzcd} & \circ & & \\ {} \arrow[r, leftrightarrow] & \circ \arrow[u, "y"] \arrow[r, "x"] & \circ \arrow[r, "x"] & \circ \end{tikzcd}$$

From this diagram we can see immediately that $(yy^{-1}x^2x^{-2})\rho = (x^2x^{-2}yy^{-1})\rho$ and also that $(yy^{-1}x^2x^{-2})\rho$ is an idempotent.

Now we are ready to state a very important result.

Theorem 3.2.14 (Munn). *The word problem for $\mathbf{F}_{\mathcal{IM}}(X)$ is solvable.*

PROOF: It is easy to see that there is an algorithm for deciding whether or not two given elements of $\mathcal{BT}_X$ are isomorphic. Let $u, v \in \mathbf{F}_{\mathcal{IM}(X)}$. Construct (α_u, T_u, β_u) and (α_v, T_v, β_v). Then by Theorem 3.2.12, $u\rho = v\rho$ if and only if these elements of $\mathcal{BT}_X$ are identical. ■

3.3 Free Group Representations

Having established the link between birooted word trees and the free inverse monoid, we can now introduce a short hand notation for a birooted word tree. This representation is found in Petrich [18]. As we shall see it is very similar to Munn's birooted word tree representation, but less cumbersome especially from the point of view of computation.

NOTATION: Let $u \in (X \cup X^{-1})^*$, $\overline{u}$ is the standard unique normal form for u in $\mathbf{F}_{\mathcal{G}}(X)$ that is, it is the *free cancellation* of u.

Definition 3.3.1. Let $u \in \mathbf{F}_{\mathcal{IM}}(X)$ with $u = x_1x_2...x_n$ where $x_1, x_2, ..., x_n \in X \cup X^{-1}$. Then the *free group representation* for u denoted by $FG(u)$ is the double (W, w) where $W = \{\epsilon, \overline{x_1}, \overline{x_1x_2}, \overline{x_1x_2x_3}, ..., \overline{u}\}$ and $w = \overline{u}$.

As free cancellation in free groups provides a unique normal form for groups we know that there is exactly one free group representation for every word $u \in (X \cup X^{-1})^*$ and so free group representations are well and uniquely defined.

Lemma 3.3.2. *Let $u, v \in \mathbf{F}_{\mathcal{IM}}(X)$ then $u\rho = v\rho$ if and only if $FG(u) = FG(v)$.*

PROOF: Suppose that $u\rho = v\rho$.

We know from Theorem 3.2.12 that there exists a unique (up to isomorphism) birooted word tree (α_u, T_u, β_u) such that for any spanning (α_u, β_u)-walk P, $w(P)\rho = u\rho$. We construct (W, w) as follows:

$$W = \{\overline{w(Q)} | Q \text{ is an } (\alpha, \gamma)\text{-walk for each } \gamma \in V(T)\},$$

$$w = \overline{w(P)} \text{ for some spanning } (\alpha_u, \beta_u)\text{-walk } P.$$

Note that w is uniquely defined because by Lemma 3.2.8 any two spanning (α_u, β_u)-walks are Wagner equivalent and are hence equal after free cancellation.

Now supposing that $u = x_1x_2...x_n$, let $u_m = x_1x_2...x_m$ with corresponding birooted tree $(\alpha_{u_m}, T_{u_m}, \beta_{u_m})$ for each $(m \leq n)$. By Theorem 3.2.13 T_{u_m} can be embedded in T_u. Noting that for any two (α, γ)-walks Q and Q' that $\overline{w(Q)} = \overline{w(Q')}$ we have

$$W = \{\overline{x_1x_2...x_m} | 0 \leq m \leq n\}$$

and so $(W, w) = FG(u)$.

By Theorem 3.2.12 $(\alpha_u, T_u, \beta_u) \cong (\alpha_v, T_v, \beta_v)$ and so the same arguement applies to the (α_v, T_v, β_v) as it did with the (α_u, T_u, β_u) above hence $(W, w) = FG(v)$ and so $FG(u) = FG(v)$ as required.

Suppose that $FG(u) = FG(v) = (W, w)$. Let $u = x_1x_2...x_n$ where $x_i \in X \cup X^{-1}$.

I proceed by constructing the birooted word tree (α, T, β) by constructing the sequence of subtrees $T_0, T_1, ..., T_n$ of T with $T = T_n$. We define T_0 to be the word tree with a single vertex we label this vertex ϵ. Given the tree T_i we define T_{i+1} in one of the two following ways:

1. If $\overline{x_1x_2...x_i} = \overline{x_1x_2...x_j}$ for some $0 \leq j \leq i$ then we define $T_{i+1} = T_i$.
2. If $\overline{x_1x_2...x_i} \neq \overline{x_1x_2...x_j}$ for each $0 \leq j \leq i$ then we adjoin a new vertex to T_i such that there is an edge labelled x_i from this new vertex to the vertex labelled $\overline{x_1x_2...x_i}$. Label this new vertex $\overline{x_1x_2...x_{i+1}}$.

At the ith step of this procedure we need to show that T_i has a vertex labelled by $\overline{x_1x_2...x_j}$ for each $1 \leq j \leq i$. Clearly T_0 has this property. Suppose T_k has the above property for all $k < i$. If the ith step is an example of case 1. then it is clear that T_i also has the above property.

For case 2. we note that $\overline{x_1x_2...x_i} = \overline{x_1x_2...x_{i-1}}x_i$. By hypothesis we need only check the new vertex. Now look at the smallest j where $\overline{x_1x_2...x_{i-1}} = \overline{x_1x_2...x_j}$. Then T_j has the required property by our hypothesis and therefore there is a vertex labelled by $\overline{x_1x_2...x_j}$ as we required in T_i. Hence by induction on i and noting the similarities of this procedure with the construction of (α_u, T_u, β_u) in Lemma 3.2.9 we can see that $(\alpha, T, \beta) \cong (\alpha_u, T_u, \beta_u)$ where α is the vertex labelled by ϵ and β is the vertex labelled by $\overline{u}$.

Similarly we can use the same induction proof to show that (α, T, β) is in fact isomorphic to (α_v, T_v, β_v). Hence by Theorem 3.2.12 we know that $u\rho = v\rho$. ■

Corollary 3.3.3. *For $u \in \mathbf{F}_{\mathcal{IM}}(X)$ and corresponding free group representation (W, w), we can construct $(\alpha, T, \beta) \in \mathcal{BT}$ such that for any (α, β)-walk P, $w(P) = \overline{u} = w$. If P is a spanning (α, β)-walk then $w(P)\rho = u\rho$.*

NOTATION: Let $\mathcal{FG}_X$ denote the class of free group representations over X.

By Theorem 3.2.12 and Corollary 3.3.3 there is a one-to-one correspondence between elements of $\mathbf{F}_{\mathcal{IM}}(X)$, $\mathcal{BT}_X$ and $\mathcal{FG}_X$

EXAMPLE: Given the word $x^2x^{-1}y^2y^{-1} \in \mathbf{F}_{\mathcal{IM}}(\{x,y\})$ then the free group representation is $(\{\epsilon, x, x^2, xy, xy^2\}, xy)$ and the corresponding birooted word tree is:

$$\begin{array}{ccccccc}
 & & & & \circ & & \\
 & & & & \uparrow_y & & \\
 & & & & \circ & \rightarrow & \\
 & & & & \uparrow_y & & \\
\rightarrow & \circ & \rightarrow_x & & \circ & \rightarrow_x & \circ
\end{array}$$

We may talk about $\mathcal{W}$*-classes* where $u\mathcal{W}v$ if and only if $FG(u) = (W, \overline{u})$ and $FG(v) = (W, \overline{v})$.

Lemma 3.3.4. *If $u \in \mathbf{F}_{\mathcal{IM}}(X)$ with $FG(u) = (W_u, w_u)$ then $FG(uu^{-1}) = (W_u, \epsilon)$.*

PROOF: Suppose that $u = x_1x_2...x_n$ where $x_i \in X \cup X^{-1}$ and $FG(u) = (W_u, w_u)$. Then $uu^{-1} = x_1x_2...x_n{x_n}^{-1}{x_{n-1}}^{-1}...{x_1}^{-1}$, define $(W_{uu^{-1}}, w_{uu^{-1}}) = FG(uu^{-1})$. Let $v \in W_{uu^{-1}}$ now either $v = \overline{x_1x_2...x_i}$ for some $0 \leq i \leq n$ or $v = \overline{x_1x_2...x_n{x_n}^{-1}{x_{n-1}}^{-1}...{x_j}^{-1}}$ for some $0 \leq j \leq n$. In the former case clearly $v \in W_u$, in the latter case $v = \overline{x_1x_2...x_{j-1}} \in W_u$. Clearly $W_u \subseteq W_{uu^{-1}}$ and so $W_{uu^{-1}} = W_u$. Clearly $\overline{uu^{-1}} = \epsilon$ and so $FG(uu^{-1}) = (W_u, \epsilon)$ as required. ■

Theorem 3.3.5. *Given any $u, v \in (X \cup X^{-1})^*$ then $u\mathcal{W}v$ if and only if $u\rho\mathcal{R}v\rho$.*

PROOF: Suppose that $u\mathcal{W}v$ in $\mathbf{F}_{\mathcal{IM}}(X)$. Now by Lemma 3.3.4 $FG(uu^{-1}) = (W, \epsilon) = FG(vv^{-1})$ for some set W that is $uu^{-1}\rho = vv^{-1}\rho$. Hence

$$(u(u^{-1}v))\rho = ((uu^{-1})v)\rho = ((vv^{-1})v)\rho = v\rho$$

and

$$(v(v^{-1}u))\rho = ((vv^{-1})u)\rho = ((uu^{-1})u)\rho = u\rho,$$

so $u\rho\mathcal{R}v\rho$.

Conversely suppose that $u\rho\mathcal{R}v\rho$ in $\mathbf{F}_{\mathcal{IM}}(X)$. By Theorem 2.3.2 we know that $u\rho\mathcal{R}uu^{-1}\rho = vv^{-1}\rho\mathcal{R}v\rho$. Hence $FG(u) = (W_u, w_u)$ where $FG(uu^{-1}) = (W_u, \epsilon)$ and $FG(v) = (W_v, w_v)$ where $FG(vv^{-1}) = (W_v, \epsilon)$, but $uu^{-1}\rho = vv^{-1}\rho$ and so $W_u = W_v$. Hence $u\mathcal{W}v$ as required. ■

To conclude we can note that $\mathbf{F}_{\mathcal{IM}}(X)$ can be regarded as a collection of birooted trees, which are actually subsets of the Cayley graph for groups. The important point is that given $u, v \in (X \cup X^{-1})^*$ with $(uu^{-1}v)\rho\mathcal{R}(uu^{-1})\rho$ then $(uu^{-1}vv^{-1})\rho = (uu^{-1})\rho$. That is, at a local $\mathcal{R}$-class level, the multiplication on the right by an inverse behaves as if it would do in $\mathbf{F}_{\mathcal{G}}(X)$. If you like you might think of $\mathbf{F}_{\mathcal{IM}}(X)$ as being in a sense "locally $\mathbf{F}_{\mathcal{G}}(X)$".

3.4 An Attempt at Enumerating Inverse Monoids

We now understand $\mathbf{F}_{\mathcal{IM}}(X)$ well enough to be able to enumerate it in the same way as we did with $\mathbf{F}_{\mathcal{SL}^1}(X)$ in **Section 3.1**. However this is a fruitless exercise as it is always infinite (except when X is empty). The question is whether it is possible to combine the technique of pushing a row of a relation with the technique of finding the unique normal form for words in the free object. Certainly there are situations where these two techniques can not be combined, for example any inverse monoid with unsolvable word problem.

Essentially the problem is not knowing how the implicit relations are affected by the explicit relations. An explicit relation, (u, v) will affect all implicit relations with either u or v as a subword on either their left or right hand sides. In the case of inverse monoids (with the standard implicit relations) adding a single explicit relation will *always* affect an infinite number of explicit relations. However with groups this is not the case, indeed with groups adding explicit relations is very simple, because not only are there just $2n$ implicit relations (where n is the number of generators) but all of these relations are very simple and have the identity as their right hand sides.

Having said all this it is possible to enumerate individual inverse monoids by taking careful account of $\mathcal{R}$-classes using a generalised notion of free group representatives. Again, the important point is that in any inverse semigroup, S, with $u, v \in S$, if $uu^{-1}\mathcal{R}uu^{-1}v$ then $uu^{-1}vv^{-1} = uu^{-1}$ (see **Chapter 4** for more about this sort of thing). In other words inverse semigroups "behave like groups" with respect to right multiplication within $\mathcal{R}$-classes. What is needed is a test for when $uu^{-1}v\mathcal{R}uu^{-1}$, that is if $v = x_1x_2...x_n$ we need to know the unique value of $1 \le i \le n-1$ such that $uu^{-1}x_1x_2...x_i \in R_u$ and $uu^{-1}x_1x_2...x_{i+1} \notin R_u$.

Given an inverse monoid presentation $\langle X|U\rangle$ for M, given a relation $(u, v) \in U$ and an element $w \in M$ then if there exists $z \in R_w$ such that $zu \in R_w$ then $zv' \in R_w$ for any v' with $v's = v$. As shall be made explicit in **Chapter 4**, this is much like tracing a relation in group Todd-Coxeter.

Hence given the inverse monoid $M = (X \cup X^{-1})^*/\tau$ what I wish to do is expand the notion of free group representatives so that it consists of pairs (W, w) where $W = \{w_1, w_2, ..., w_n\}$ contains freely reduced words such that

$$\{(ww^{-1}w_1)\tau, (ww^{-1}w_2)\tau, ..., (ww^{-1}w_n)\tau\} = R_{w\tau}.$$

With this concept of free group representatives there is, however, no guarantee that $(ww^{-1}w_i)\tau \neq (ww^{-1}w_j)\tau$ when $i \neq j$.

This is roughly the reasoning behind my first attempt at coding an inverse monoid enumerator. This algorithm is new but is based on the group and monoid Todd-Coxeter algorithm's in *Section 1.3* and *Section1.5*. I shall give a rough description of it before moving on to a more general understanding of $\mathcal{R}$-classes in **Chapter 4** and a much improved enumerator which borrows some ideas from Nik Ruškuc, Allessandra Cherubini and Brunnetto Piochi in **Chapter 5**.

3.4.1 The Data Structures

- The immutable finite inverse monoid *presentation* $P = \langle X | U \rangle$.
- The mutable *set of cosets* C which is a finite set of positive integers which initially is $C = \{1\}$.
- The mutable *coset table* T whose entries are elements of $C \cup \{\perp\}$ where $\perp$ is the empty symbol and whose columns are labelled by elements of $X \cup X^{-1}$ and whose rows are labelled by elements of C. I shall refer to the entry in row c and column x by $T(c, x)$. Initially T contains a single empty row of $\perp$'s.
- The mutable *truth coset table* T' whose entries are elements of $\{0, 1, \perp\}$ and whose columns are labelled by elements of $X \cup X^{-1}$ and whose rows are labelled by elements of C. I shall refer to the entry in the row c and column x by $T'(c, x)$. Initially T' contains a single empty row of $\perp$'s.
- The mutable *free group representative set* Φ contains elements of the form (W, w) where $W \subseteq (X \cup X^{-1})^*$ and $w \in W$. Initially $\Phi := \{(\{\epsilon\}, \epsilon)\}$.
- A surjection $\psi : C \to \Phi$ with $1\psi = (\{\epsilon\}, \epsilon)$.
- The mutable *coincidence set* $K \subseteq C \times C$. Initially K is empty.

- The mutable *representative coincidence set* $\kappa \subseteq \Phi \times \Phi$. Initially κ is empty.
- An equivalence $\sim$ on C and also $D \subseteq C$ of elements which need to be forced to be $\sim$ related. Initially D is empty.

Clearly there are many more data structures than there were for the standard monoid enumeration algorithm. In particular we have Φ, ψ and κ all dealing with representatives all of which must be processed while the relations in U are processed. In addition it is necessary to record which elements of C are in which $\mathcal{R}$-class, which is the purpose of $\sim$. Even then the algorithm is, as we shall see, flawed as it cannot immediately distinguish between different $\mathcal{R}$-classes.

3.4.2 The Subroutines

Before I describe these it is important to understand how $\mathcal{R}$-classes work in this algorithm. Given $c \in C$ and $x \in X \cup X^{-1}$ suppose that $T'(c, x) = 1$. This means that the element $u \in M$ represented by the coset c is $\mathcal{R}$ related to ux (it is possible that $u = ux$). The algorithm will occasionally update 0 entries in T' to 1's when it discovers that two $\mathcal{R}$-classes are identical therefore a 0 entry merely means that "u and ux are not $\mathcal{R}$-related as far as we know."

Similarly an element $(W, w) \in \Phi$ will constantly be updated, but only by adding to and not taking away from or changing elements of W as new words are discovered to be $\mathcal{R}$ related to w. As a consequence at all stages

$$\{\epsilon, x_1, \overline{x_1 x_2}, ..., \overline{x_1 x_2 ... x_n}\} \subseteq W$$

when $w = \overline{x_1 x_2 ... x_n}$. It is important to note that, unlike the free inverse monoid case, different elements in W do not necessarily represent different elements in R_w. For example $ww^{-1}x_1x_2...x_i = ww^{-1}x_1x_2...x_j$ (for some $1 \leq i < j \leq n$) might be a consequence of U and so $x_1...x_i$ and $x_1...x_j$ will represent the same element in R_w even though they are both different elements in W.

Create a *New* Definition

- Parameters: $c \in C$, $x \in X \cup X^{-1}$.
- Local: d

This subroutine is much like the same in the monoids algorithm except for the following points

- A new empty row is added to T'
- Suppose that $c\psi = (W, w)$ then add $(W \cup \{wx\}, wx)$ to Φ and if the new coset is d let $d\psi := (W \cup \{wx\}, wx)$.
- If $wx \in W$ then let $c \sim d$, let $T'(c, x) := 1$ and let $T(d, x^{-1}) := c$.
- If $wx \notin W$ then let $c \not\sim d$ and let $T'(c, x) := 0$.

Update $\sim$

- Parameter: $E \subseteq C$
- Locals: e, f, x

This is a "book keeping subroutine" which forces all the elements of E to be $\sim$ related. Suppose that for $e \in E$ that $e\psi = (W_e, w_e)$, then we update $e\psi := (\cup_{f \in E} W_f, w_e)$. Also if $e, f \in E$ such that there exists $x \in X$ with $T(e, x) = f$ then $T'(e, x) := 1$. Finally E is added to D.

Update Φ

- Parameters: None
- Locals: d, e, x

Another "book keeping subroutine" which looks at the set D of cosets which have had their free group representatives recently modified. This subroutine takes each $d \in D$ where $d\psi = (W_d, w_d)$ and each adjacent coset $e = T(d, x)$ where $e\psi = (W_e, w_e)$ and sets $e\psi := (W_e \cup W_d, w_e)$.

Trace **a Relation**

- Parameters: $c \in C$ and $(u, v) \in U$
- Locals: k, l, m

Suppose that $u = u_1u_2...u_m$ and that $v = v_1v_2...v_n$. This subroutine compares $T(c,u)$ with $T(c,v)$ and $T'(c,u)$ with $T'(c,v)$. There are eight cases where changes need to be made, although we only need to discuss four of them as the other four are symmetric in u and v.

1. $T(c,u) = \perp, T(c,u_1...u_{m-1}) = k \neq \perp, T(c,v) = l \neq \perp$ and $T'(c,v) = 0$. In this case we define $T(k,u_m) := l$. Furthermore if $T'(c,u_1...u_{m-1}) = 1$ then set $T'(l,u_m) := T'(c,v)$ and then apply the subroutine **Update** $\sim (\{T(c,u),k\})$.

   Note that the case where $T(c,u_1u_2...u_{m-1}) = \perp$ tells us nothing. It is only the case where the relations *force* conclusions in the coset table that tell us anything.

   Note also that it is not necessary to check that $T'(c,u_1...u_{m-1}) = 0$ as although $\mathcal{R}$ is transitive, the failure to be $\mathcal{R}$ related is not transitive, ie. if $r \notin R_s$ and $s \notin R_t$ then it does not necessarily follow that $r \notin R_t$.

2. $T(c,u) = T(c,u_1...u_{m-1}) = \perp, T(c,v) = l \neq \perp$ and $T'(c,v) = 1$. Unlike in the first case, here we are forced to conclude that $c \sim T(c,u)$ as $c \sim T(c,v)$. We therefore apply **Update** $\sim (\{T(c,u_1...u_i) | 1 \leq i \leq m\})$.

3. $T(c,u) = \perp, T(c,u_1...u_{m-1}) = k \neq \perp, T(c,v) = l \neq \perp$ and $T'(c,v) = 1$. As in the first case set $T(k,u_m) := l$ and like the second case apply **Update** $\sim (\{T(c,u_1...u_i) | 1 \leq i \leq m\})$.

4. $T(c,u) = k$ and $T(c,v) = l$. Here we have a coincidence as we would in the monoid algorithm. We add (k,l) to K. Furthermore if

   (a) $T'(c,u) = T'(c,v) = 1$ then add $(k\psi, l\psi)$ to κ.

   (b) $T'(c,u) \neq 1$ and $T'(c,v) = 1$ then apply **Update** $\sim (\{T(c,u_1...u_i) | 1 \leq i \leq m\})$.

Finally **Update** Φ is applied.

Identify κ **Coincidences**

- Parameters: None
- Locals: c, d, U, u, V, v

This algorithm runs through each $((U, u), (V, v)) \in \kappa$. If a particular $(U, u) = (V, v)$ then simply ignore this one and remove from κ. Otherwise choose $c \in C$ such that $c\psi = (U, u)$ and do the following:

- For every $d \in C$ such that $d\psi = (V, w)$ with $w \in V \setminus \{v\}$, replace (V, w) by $(U \cup V, w)$ in Φ and let $d \sim c$.
- For every $d \in C$ such that $d\psi = (V, v)$ replace (V, v) by $(U \cup V, v)$ in Φ and redefine $c\psi := (U \cup V, v)$ and let $d \sim c$.
- For every $d \in C$ such that $d\psi = (U, w)$ with $w \in U \setminus \{u\}$, replace (U, w) by $(U \cup V, w)$ in Φ and let $d \sim c$.
- For every $d \in C$ such that $d\psi = (U, u)$ replace (U, u) by $(U \cup V, u)$ in Φ and redefine $c\psi := (U \cup V, v)$ and let $d \sim c$.

Identify K **Coincidences**

- Parameters: None
- Locals: c, d, x

This is similar to the monoid algorithm **Identify** subroutine. Given $(c, d) \in K$ then the following differences are noted.

1. If for some $x \in X \cup X^{-1}$, $T'(d, x) = 1$ and if $T'(c, x) \neq 1$ then redefine $T'(c, x) := 1$ and apply **Update** $\Phi(\{c, T(c, x)\})$.
2. **Update** $\Phi(\{c, d\})$ is applied.
3. d is replaced by c in all the data structures.

The Main Procedure

For $c \in C$ and $x \in X \cup X^{-1}$ do the following

- **New**(c, x)
- For $d \in C$ and $(u, v) \in U$, do **Trace**$(c, (u, v))$

- **Update** Φ
- **Identify** κ
- **Identify** K

Tidy Up

3.4.3 Comments

The first thing to notice is that this algorithm is quite convoluted. There is a lot of "book keeping". Secondly it is horribly inefficient with memory as elements in Φ tend to grow uncontrollably and apparently unnecessarily large. Thirdly it only *systematically* enumerates cosets within particular $\mathcal{R}$-classes, there is no guarantee that two new cosets in apparently different $\mathcal{R}$-classes are not actually the same coset in the same $\mathcal{R}$-class.

It is, however, possible to fix this last point by finding an $\mathcal{R}$-class test and eliminating excess $\mathcal{R}$-classes using some sort of "**Identify** $\mathcal{R}$**-classes**" subroutine. All this, however, points in the direction of enumerating $\mathcal{R}$-classes individually rather than simultaneously. Indeed this is what I do in **Chapter 5**.

EXAMPLE: A very simple example is the inverse monoid M presented by $\langle x | x^3 = x \rangle$. It is not hard to see that this is the cyclic group of order 2 with an extra identity attached. This can be seen by the fact that in M, x is its own inverse because $x(x)x = x^3 = x$ and by uniqueness of inverses $x^{-1} = x$ and so it is not difficult to see that M is presented by $\langle x | x^3 = x \rangle$ as a monoid.

The algorithm given above will produce the table below.

Cosets	x	x^{-1}	Φ
1	2_0	3_0	$(\{\epsilon\}, \epsilon)$
2	4_1	4_1	$(\{\epsilon, x, x^2, x^3\}, x)$
3	5_1	5_1	$(\{\epsilon, x^{-1}, x^{-2}\}, x^{-1})$
4	2_1	2_1	$(\{\epsilon, x, x^2, x^3\}, x)$
5	3_1	2_1	$(\{\epsilon, x^{-1}, x^{-2}\}, x^{-1})$

In this table the truth coset table is represented by suffixes in the entries. We also have three $\sim$-classes - $\{1\}$, $\{2, 4\}$ and $\{3, 5\}$.

Now apparently M is not a cyclic group with an extra identity but rather *two* cyclic groups with an extra identity. This is precisely because Todd-Coxeter style algorithms have no way of recognising the condition "uniqueness of inverses" and so for M seem to think that x and x^{-1} generate two different $\mathcal{R}$-classes even though $x = x^{-1}$. I believe this is fundamental problem that can only be solved by recognising the equivalence of two $\mathcal{R}$-classes *after* they have been enumerated.

I conclude by saying that before we can proceed we need a clearer understanding of inverse monoid presentations and their effect on $\mathcal{R}$-classes.

Chapter 4

The Word Problem for Inverse Monoids

As we have seen the strategies of **Chapter 3** are perhaps insightful but not adequate for the job of a Todd-Coxeter style enumeration for inverse monoids. However, as I have pointed out, if the word problem for a semigroup is solvable then there should be a strategy for enumerating its elements. In **Chapter 3** we looked at Munn's solution to the word problem for free inverse monoids. Here we shall review J. B. Stephen's paper *Presentations of Inverse Monoids* [27] extending the ideas of **Chapter 3** to general inverse monoids. Most results in this section are attributed to Stephen and the ones which are not are generalisations of Stephen's results involving right congruences.

4.1 Inverse Word Graphs and Automata

First we consider the generalisation of word trees to *word graphs*.

Definition 4.1.1. A *word graph* Γ on the set X is a connected, directed graph with edges labelled by elements of X. If α and β are adjacent vertices in Γ, then the edge labelled

by $x \in X$ from α to β is denoted (α, x, β). We shall denote the vertices of Γ by $V(\Gamma)$ and we shall denote the edges of Γ by $E(\Gamma)$. The word graph Γ is said to be finite if both $V(\Gamma)$ and $E(\Gamma)$ are finite.

Definition 4.1.2. Given two vertices α and β on a word graph Γ, the notions of (α, β)-walk on Γ, spanning (α, β)-walk on Γ and (α, β)-path are defined in the same way as for word trees (see Definition 3.2.2). The notions of *word graph homomorphism, word graph monomorphism, word graph isomorphism, word graph automorphism* are also defined in the same way as word tree homomorphisms etc. (see Definition 3.2.3).

Definition 4.1.3. A word graph is said to be *deterministic* if all the edges directed from a vertex are labeled by different letters and it said to be *injective* if all edges directed towards a vertex are labeled by different letters. In other words, a deterministic, injective word graph obeys (WT2) (see Definition 3.2.1).

Definition 4.1.4. An *inverse word graph* over X is a connected, directed graph, Γ, with edges labeled by elements from $X \cup X^{-1}$ in such a way that the labeling is consistent with involution; that is $(\gamma, x, \delta) \in E(\Gamma)$ if and only if $(\delta, x^{-1}, \gamma) \in E(\Gamma)$, where $x \in X \cup X^{-1}$ and $\gamma, \delta \in V(\Gamma)$. For convenience we shall assume that at every vertex of Γ there is a loop labeled by ϵ.

A *birooted inverse word graph* is a triple $A = (\alpha, \Gamma, \beta)$ where Γ is an inverse word graph over X with $\alpha, \beta \in V(\Gamma)$. We call α the *start* of Γ and we call β the *end* of Γ.

The notion of word graph homomorphism extends to birooted inverse word graphs. A *birooted inverse word graph homomorphism* $\phi : (\alpha, \Gamma, \beta) \to (\alpha', \Gamma', \beta')$ is a word graph homomorphism from Γ to Γ' with the special conditions $\alpha\phi = \alpha'$ and $\beta\phi = \beta'$. Similarly we have extensions for word graph monomorphism, word graph epimorphisms and word graph automorphisms. I will call birooted inverse word graph homomorphisms word graph homomorphisms or simply homomorphisms where there is no confusion.

Lemma 4.1.5. *An inverse word graph over $X \cup X^{-1}$ is deterministic if and only if it is injective.*

PROOF: This is clear from Definition 4.1.4. ■

It should be noted that a birooted inverse word graph $A = (\alpha, \Gamma, \beta)$ is also an automaton see, for example, John Howie's *Automata and Languages* [10]. In particular A has the following features:

(i) As each edge of A has an inverse edge, A is strongly connected and is therefore *trim*

(ii) A has only one *terminal state* - β. It is a *one-out automaton.* The *initial state* of A is α.

(iii) A has a special inverse property as described in Definition 4.1.4. It is an *inverse automaton*

(iv) If Γ is deterministic then A is a *deterministic automaton.*

NOTATION: If $A = (\alpha, \Gamma, \beta)$ is an automaton over X, and if $w \in X^*$ traces a (γ_1, γ_2)-walk in Γ then we say $\gamma_1 w = \gamma_2$. The set $L[A] = \{w \in X^* | \alpha w = \beta\}$ is the *language recognised by A.*

So far we have looked at ideas which were discussed in a more specific form (ie. with respect to word trees) in **Section 3.2**. However, homomorphisms between word trees are always one-to-one maps on the vertices. With word graphs we have a richer theory.

Definition 4.1.6. Given a word graph Γ over X and an equivalence relation η on $V(\Gamma)$ the *quotient of* Γ *induced by* η is the word graph denoted by Γ/η. The vertices of Γ/η are the equivalence classes of $V(\Gamma)/\eta$. The edges of Γ/η are given by:

$$E(\Gamma/\eta) = \{(\gamma_1/\eta, x, \gamma_2/\eta) | (\gamma_1, x, \gamma_2) \in E(\Gamma)\}$$

We have a first isomorphism theorem for inverse word graphs.

Theorem 4.1.7. *Given word graphs Γ and Δ over X and an epimorphism $\phi : \Gamma \to \Delta$ then Δ is isomorphic to Γ/η for some equivalence η on $V(\Gamma)$.*

Furthermore we have the following important lemma:

Lemma 4.1.8 (Stephen). *Let Γ and Γ' be word graphs and $\phi : \Gamma \to \Gamma'$ be a homomorphism and let $\alpha, \beta \in V(\Gamma)$. If w labels an (α, β)-walk in Γ, then w labels an $(\alpha\phi, \beta\phi)$-walk in Γ'. So if $A = (\alpha, \Gamma, \beta)$ and $A' = (\alpha\phi, \Gamma', \beta\phi)$, then $L[A] \subseteq L[A']$.*

PROOF: This follows straight from Definition 4.1.1. ∎

The converse of this does not always follow unless we are dealing with deterministic inverse automata as shown in the following theorem.

Theorem 4.1.9 (Stephen). *Let $A = (\alpha, \Gamma, \beta)$ be a birooted word graph over $X \cup X^{-1}$ and let $A' = (\alpha', \Gamma', \beta')$ be a deterministic birooted inverse word graph over $X \cup X^{-1}$. If $L[A] \subseteq L[A']$, then there is a homomorphism $\phi : \Gamma \to \Gamma'$ such that $\alpha\phi = \alpha'$ and $\beta\phi = \beta'$.*

PROOF: We define the map ϕ on the vertices of Γ as follows. If $\gamma \in V(\Gamma)$, then, since A is strongly connected, there is a $w \in (X \cup X^{-1})^*$ such that $\alpha w = \gamma$. Define $\gamma\phi$ to be $\alpha' w$. It is necessary to show that ϕ is well defined.

Suppose $\alpha w = \gamma$ and $\alpha w' = \gamma$, now note that since A is strongly connected there is a $u \in (X \cup X^{-1})^*$ such that $\gamma u = \beta$. Thus $wu, w'u \in L[A] \subseteq L[A']$. We wish to show that $\alpha' w = \alpha' w'$, so note that $\alpha' w = \beta' u^{-1}$ and $\alpha' w' = \beta' u^{-1}$, since A' is deterministic and inverse. Thus $\alpha' w = \alpha' w'$, and ϕ is well defined on the vertex set.

Suppose $(\gamma_1, x, \gamma_2) \in E(\Gamma)$, then there exists $w_1, w_2 \in (X \cup X^{-1})^*$ such that w_1 labels an (α, γ_1)-path and w_2 labels a (γ_2, β)-path in Γ. Now $w_1 x w_2$ labels an α', β'-walk in Γ', so $(\gamma_1\phi, x, \gamma_2\phi) \in E(\Gamma')$. Thus we see that ϕ is a homomorphism. ■

Now there are many automata which recognise the same language L. We want to pick a particular automaton - the *minimal automaton* - which is unique for each language. We have the following result for automata (see Reutenauer [20] for the definition of minimal and for the proof):

Lemma 4.1.10 (Reutenauer). *Let $A = (\alpha, \Gamma, \beta)$ be a trim one-out automaton. If Γ is deterministic and injective, then A is the minimal automaton accepting $L(A)$.*

The following lemma is an immediate consequence of Lemmas 4.1.5 and 4.1.10.

Lemma 4.1.11. *If Γ is a deterministic inverse word graph over $X \cup X^{-1}$, and $\alpha, \beta \in V(\Gamma)$, then (α, Γ, β) is the minimal automaton accepting $L[(\alpha, \Gamma, \beta)]$.*

As a corollary it follows that if Γ is a word tree then $A = (\alpha, \Gamma, \beta)$ is minimal as Γ is by definition deterministic.

The thinking here is that coset tables can be viewed as deterministic word graphs and so given a word $w \in (X \cup X^{-1})^*$ it is our task to find a minimal automaton A which accepts w (ie. $w \in L[A]$). We shall return to this "determinising process" after examining *Schützenberger graphs*.

4.2 Schützenberger Graphs

In this section we introduce an important concept from semigroup theory (see Petrich [18] or Howie [9]).

For the rest of this section M is an inverse monoid with presentation $\langle X|U\rangle$, with τ being the congruence generated by $\rho \cup U$ (ρ being the Wagner congruence for $\mathbf{F}_{\mathcal{IM}}(X)$). If $u \in M$ then H_u, R_u, L_u and D_u are the Green's equivalence classes of u. We let σ be the minimal group congruence. Recall that for $u, v \in M$, we write $u \geq v$ if there exists an idempotent $e \in M$ such that $ue = v$. This last is equivalent to saying that $u \geq v$ if there exists an idempotent $f = ueu^{-1} \in M$ such that $fu = v$.

Definition 4.2.1. Let $u \in M$. The *Schützenberger graph* of R_u is the word graph $S\Gamma(u)$ where

$$V(S\Gamma(u)) = R_u$$

and

$$E(S\Gamma(u)) = \{(v_1, x, v_2)|v_1, v_2 \in R_u, x \in X \cup X^{-1}, v_1(x\tau) = v_2\}.$$

NOTATION: Let A be a set and let $B \subseteq A$. We define the equivalence relation on A,

$$|_B^A = \{(b_1, b_2)|b_1, b_2 \in B\} \cup \{(a_1, a_2)|a_1, a_2 \in A \setminus B\}.$$

Definition 4.2.2. Let $u \in M$ and let ζ be a right congruence on M such that

$$\zeta \subseteq |_{R_u}^M.$$

Then the set R_u/ζ is the set of all equivalence classes of ζ restricted to R_u. The *right quotient of* $S\Gamma(u)$ *by* ζ is the word graph $S\Gamma(u)/\zeta$ where

$$V(S\Gamma(u)/\zeta) = R_u/\zeta$$

and

$$E(S\Gamma(u)/\zeta) = \{(v_1, x, v_2)|v_1, v_2 \in R_u/\zeta, x \in X \cup X^{-1}, v_1((x\tau)\zeta) = v_2\}$$

The condition that $\zeta \subseteq |^M_{R_u}$ is necessary in the above definition to make R_u/ζ well defined as otherwise ζ is not an equivalence relation on R_u.

I introduce the idea of right quotients of Schützenberger graphs because they are useful for simplifying the structures and we need to show to what extent they are consistent with Stephen's theory for Schützenberger graphs. Of course the special case of proper quotients (quotients which are both right quotients and left quotients) in effect add new relations to U and thus can be thought of as the $\mathcal{R}$-class on a new inverse monoid. It is useful to recall that $\mathcal{L}$ is a right congruence while $\mathcal{R}$ is a left congruence.

It is important to note the distinction between right congruences on M and right congruences on $S\Gamma(u)$. We can think of the latter as "local right congruences". It could be a fruitful area of research to look at what exactly we can do with right congruences on M which fail to satisfy the condition that $\zeta \subseteq |^M_{R_u}$. It is however difficult to even define the Schützenberger graph in this case as R_u will be identified with R_v by ζ for some $v \notin R_u$.

Definition 4.2.3. There is also the dual concept of *left Schützenberger graph* (and a corresponding notion of *left quotients*), $S\Gamma^1(u)$, of L_u where

$$V(S\Gamma^1(u)) = L_u$$

and

$$E(S\Gamma^1(u)) = \{(v_1, x, v_2) | v_1, v_2 \in L_u, x \in X \cup X^{-1}, (x\tau)v_1 = v_2\}.$$

All of the results which follow for $S\Gamma(u)$ can be dualised for $S\Gamma^1(u)$.

Lemma 4.2.4. *Given $u \in M$ and a right congruence ζ on $S\Gamma(u)$, there is a word graph homomorphism $\phi : S\Gamma(u) \to S\Gamma(u)/\zeta$ so that if w labels a (v_1, v_2)-walk in $S\Gamma(u)$ then its image under ϕ labels a $(v_1\zeta, v_2\zeta)$-walk in $S\Gamma(u)/\zeta$.*

PROOF: For $v \in V(S\Gamma(u))$, define $v\phi = v\zeta$ and for $(v_1, x, v_2) \in E(S\Gamma(u))$, define $(v_1, x, v_2)\phi = (v_1\zeta, x, v_2\zeta)$. Clearly this map is well defined and preserves the labeling of edges.

Given any (v_1, v_2)-walk in $S\Gamma(u)$, which is given by $P = (v_1 = \gamma_0, \gamma_1, ..., \gamma_n = v_2)$ with P labelled by $(x_1, x_2, ..., x_n)$, then the image walk starts at $v_1\zeta$ and, as the action is on the right, each $\gamma_i = (v_1\zeta)x_1x_2...x_i$ as required. ■

Lemma 4.2.5 (Stephen). *Given $u \in M$, $S\Gamma(u)$ is a trim, inverse, deterministic word graph.*

PROOF: $S\Gamma(u)$ is trim because it is strongly connected. $S\Gamma(u)$ is deterministic since multiplication is well defined on M.

To show that $S\Gamma(u)$ is inverse consider $(v_1, x, v_2) \in E(S\Gamma(u))$. Now

$$e = v_1 v_1^{-1} = v_2 v_2^{-1} = v_1(x\tau)(x^{-1}\tau)v_1^{-1},$$

so

$$\begin{aligned} v_1 &= ev_1 = v_1(x\tau)(x^{-1}\tau)v_1^{-1}v_1 = v_1 v_1^{-1} v_1(x\tau)(x^{-1}\tau) \\ &= v_1(x\tau)(x^{-1}\tau) = v_2(x^{-1}\tau). \end{aligned}$$

Thus $(v_2, x^{-1}, v_1) \in E(S\Gamma(u))$ and so $S\Gamma(w)$ is an inverse word graph. ■

Given that $ww^{-1}\mathcal{R}w$ (see Lemma 2.2.2) we may talk about the minimal automaton $A_w = (ww^{-1}, S\Gamma(w), w)$ corresponding to w.

Theorem 4.2.6. *Let $u \in M$, let $\zeta \subseteq |_{R_u}^M$ be a right congruence on R_u, let $u_1, u_2 \in R_u$ and $w \in (X \cup X^{-1})^*$. Then $u_1(w\tau)\zeta = u_2\zeta$ if and only if w labels a $(u_1\zeta, u_2\zeta)$-walk in $S\Gamma(u)/\zeta$.*

PROOF:

The proof is by induction on $|w|$. If $|w| = 0$ that is $w = \epsilon$, then by definition of $S\Gamma(u)/\zeta$, $u_1(w\tau)\zeta = u_2\zeta$ if and only if $(u_1\zeta, \epsilon, u_2\zeta) \in E(S\Gamma(u))$, or w labels a $(u_1\zeta, u_2\zeta)$-walk, that is $u_1\zeta = u_2\zeta$ and so $u_1(w\tau)\zeta = u_2\zeta$ as required. If $w = \epsilon$ and $u_1(w\tau)\zeta = u_2\zeta$ then $u_1\zeta = u_2\zeta$ so w labels the empty $(u_1\zeta, u_2\zeta)$-walk. Now suppose that the result is true for all words of length less than N.

Let $w \in (X \cup X^{-1})^*$ such that $|w| = N$. If $w = x_1x_2...x_N$ is a $(u_1\zeta, u_2\zeta)$-walk there exists $u_2' \in R_u$ such that $w = x_1x_2...x_{N-1}$ labels a $(u_1\zeta, u_2'\zeta)$-walk. By hypothesis, $u_1(x_1x_2...x_{N-1}\tau)\zeta = u_2'\zeta$ and, since w labels a $(u_1\zeta, u_2\zeta)$-walk, $(u_2'\zeta, x_N, u_2\zeta) \in E(S\Gamma(u)/\zeta)$. By the definition of $S\Gamma(u)/\zeta$ we have $u_2\zeta = u_2'(x_N\tau)\zeta$ and so

$$u_1(w\tau)\zeta = u_1(x_1...x_{N-1}x_N\tau)\zeta = u_2'(x_N\tau)\zeta = u_2\zeta$$

as required.

Conversely, if $u_1(w\tau)\zeta = u_2\zeta$ then choose u_2' such that

$$u_1(x_1x_2...x_{N-1}\tau)\zeta = u_2'\zeta \in R_u/\zeta.$$

Then by the induction hypothesis $x_1...x_{N-1}$ labels a $(u_1\zeta, u_2'\zeta)$-walk. Now $u_1(x_1...x_{N-1}\tau)(x_N\tau)\zeta = u_2\zeta$ and so $u_2'(x_N\tau)\zeta = u_2\zeta$ and so $x_1x_2...x_N$ labels a $(u_1\zeta, u_2\zeta)$-walk.

■

For the special case of Schützenberger graphs without right congruences we have the following results found in Stephen's paper.

Corollary 4.2.7 (Stephen). *Let $u \in M$, $e = uu^{-1}$, $u_1, u_2 \in R_u$ and $w, w_1, w_2 \in (X \cup X^{-1})^*$. The following statements hold.*

(i) *$u_1(w\tau) = u_2$ if and only if w labels a (u_1, u_2)-walk in $S\Gamma(u)$.*

(ii) *$w\tau \geq u$ if and only if w labels an (e, u)-walk in $S\Gamma(u)$.*

(iii) *If w_1 and w_2 both label (u_1, u_2)-walks in $S\Gamma(u)$ then $w_1\sigma = w_2\sigma$.*

PROOF:

(i) This is Theorem 4.2.6 with ζ as the trivial right congruence.

(ii) Suppose $w\tau \geq u$, then $e(w\tau) = u$, so by letting ζ be trivial in Theorem 4.2.6, w labels an (e, u)-walk. Conversely if w labels an (e, u)-walk, then by letting ζ be trivial in Theorem 4.2.6, $e(w\tau) = u$, so $w\tau \geq u$.

(iii) Suppose that w_1 and w_2 both label (u_1, u_2)-walks, then $u_1(w_1\tau) = u_1(w_2\tau)$ and so $u_1(w_1\tau)\sigma = u_1(w_2\tau)\sigma$ and by group cancellation $w_1\sigma = w_2\sigma$.

■

It is worth noting how much results (ii) and (iii) in the above corollary generalise. The following corollary uses the same notation as Theorem 4.2.6 and Corollary 4.2.7.

Corollary 4.2.8. *If $w\tau \geq u$ then w labels an $(e\zeta, u\zeta)$-walk in $S\Gamma(u)/\zeta$.*

PROOF: Suppose $w\tau \geq u$, then $e(w\tau) = u$, so by Theorem 4.2.6, w labels an $(e\zeta, u\zeta)$-walk on $S\Gamma(u)/\zeta$ as required. ■

Of course the converse of this corollary does not hold. Even if we define a partial ordering on M/ζ so that $u\zeta \geq v\zeta$ if and only if there exists an idempotent $e \in$

M such that $(ue)\zeta = v\zeta$ then we do not get a converse as we cannot say that $(uu^{-1}(w\tau))\zeta = u\zeta$ as the left hand side of this is a *left* multiple of $w(\tau)$ and under the *right* congruence ζ it is not generally true unless uu^{-1} commutes with $w\tau$.

However it is possible to generalise Corollary 4.2.7 part (iii). The next corollary uses the same notation as in Theorem 4.2.6 and Corollary 4.2.7.

Theorem 4.2.9. *Suppose $\zeta \subseteq \sigma$. If w_1 and w_2 both label $(u_1\zeta, u_2\zeta)$-walks in $S\Gamma(u)/\zeta$ then $(w_1\tau)\sigma = (w_2\tau)\sigma$.*

PROOF: Suppose that w_1 and w_2 both label $(u_1\zeta, u_2\zeta)$-walks, then $u_1(w_1\tau)\zeta u_1(w_2\tau)$ and so $u_1(w_1\tau)\sigma u_1(w_2\tau)$ and by group cancellation $(w_1)\tau\sigma(w_2)\tau$. ■

The next lemma is simply a restatement of Green's lemma in the language of word graphs.

Lemma 4.2.10. *Let $u, v, y, z \in M$. If $yu = v$ and $zv = u$ (that is $u\mathcal{L}v$), then y and z induce mutually inverse $\mathcal{L}$-class preserving word graph isomorphisms $\phi_y : S\Gamma(u) \to S\Gamma(v)$ and $\phi_z : S\Gamma(v) \to S\Gamma(u)$ respectively, where $s\phi_y = ys$ and $t\phi_z = zt$ for $s \in V(S\Gamma(u))$ and $t \in V(S\Gamma(v))$.*

We are now ready to state the main result.

Theorem 4.2.11 (Stephen). *Let $u, v \in M$ and let $e = uu^{-1}$ and $f = vv^{-1}$. The following statements hold:*

(i) *$u\mathcal{D}v$ if and only if there exists a word graph isomorphism*

$$\phi : S\Gamma(u) \to S\Gamma(v).$$

(ii) *$u\mathcal{R}v$ if and only if there exists a word graph isomorphism*

$$\phi : S\Gamma(u) \to S\Gamma(v)$$

with the condition that $e\phi = f$.

(iii) *$u\mathcal{L}v$ if and only if there exists a word graph isomorphism*

$$\phi : S\Gamma(u) \to S\Gamma(v)$$

with the condition that $u\phi = v$.

(iv) $u\mathcal{H}v$ *if and only if there exist word graph isomorphisms*

$$\phi, \psi : S\Gamma(u) \to S\Gamma(v)$$

with the conditions that $e\phi = f$ *and* $u\psi = v$

(v) $u = v$ *if and only if there exists a word graph isomorphism*

$$\phi : S\Gamma(u) \to S\Gamma(v)$$

with the conditions that $e\phi = f$ *and* $u\phi = v$.

PROOF:

(i) Suppose that $u\mathcal{D}v$, then let $y \in R_e \cap L_f$. Now note that $y^{-1}\mathcal{L}e$ and $y^{-1}\mathcal{R}f$. It is clear that $y^{-1}e = y^{-1}$ and $y(y^{-1}) = e$, so by Lemma 4.2.10 there exists a word graph isomorphism from $S\Gamma(e) = S\Gamma(u)$ to $S\Gamma(y^{-1}) = S\Gamma(f) = S\Gamma(v)$ induced by left multiplication by y.

Conversely, if $\phi : S\Gamma(u) \to S\Gamma(v)$ is a word graph isomorphism, then let y label an $(f, e\phi)$-walk and z a $(u\phi, v)$-walk. From part (ii) of Corollary 4.2.7 it is clear that $(y^{-1}y)\tau \geq e\phi$ and so $(y^{-1}y)\tau u = u$. Thus $(y\tau)u\mathcal{L}u$. Now let $w_1 \in u\tau^{-1}$, then yw_1z labels an (f, v)-walk in $S\Gamma(v)$, so $(yw_1z)\tau \geq v$. Similarly, if $w_2 \in v\tau^{-1}$, then $(y^{-1}w_2z^{-1})\tau \geq u$. Now note $(yy^{-1}\tau)v(z^{-1}z\tau) = (y(y^{-1}w_2z^{-1})z)\tau \geq (y\tau)u(z\tau) \geq v$, but $v = w_2\tau \geq (y(y^{-1}w_2z^{-1})z)\tau$, so $(y\tau)u(z\tau) = v$. Examine the product $((y\tau)u)(z\tau)(z^{-1}\tau)$. If we can show that $u(z\tau)(z^{-1}\tau) = u$, then it will be clear that $(y\tau)u\mathcal{R}v$. To see this, note that w_1zz^{-1} labels an (e, u)-walk, so $(w_1zz^{-1})\tau = u(vv^{-1}\tau) \geq u$, but it is clear that $u \geq u(zz^{-1}\tau)$ so $u(zz^{-1}\tau) = u$. Thus $u\mathcal{D}v$.

(ii) For $u\mathcal{R}v$, let ϕ be the identity word graph isomorphism.

Conversely, if $\phi : S\Gamma(u) \to S\Gamma(v)$ is a word graph isomorphism such that $e\phi = f$, then note that the proof of the indirect implication in (i) will carry over to this case. Let $y = \epsilon$, and then note that $(y\tau)u\mathcal{R}v$ yields $u\mathcal{R}v$.

(iii) Similar to (ii) except let $z = \epsilon$ from the proof of (i).

(iv) Clear from the definition of $\mathcal{H}$ and (ii) and (iii).

(v) The direct implication is obvious. If $\phi : S\Gamma(u) \to S\Gamma(v)$ is a word graph isomorphism such that $e\phi = f$ and $u\phi = v$, then $L[(e, S\Gamma(u), u)] = L[(f, S\Gamma(v), v)]$, by Lemma 4.1.8. Now note that if $w_1 \in u\tau^{-1}$ and $w_2 \in v\tau^{-1}$, then $w_1\tau \geq v$ and $w_2\tau \geq u$ by part (ii) of Corollary 4.2.7 and so $u = v$.

■

We now turn our attention to the right quotients again. We have a result similar to Green's lemma.

Lemma 4.2.12. *Let $u, v, y, z \in M$, let $\zeta \subseteq |^M_{R_u}$ be a right congruence on M. Suppose that $yu = v$ and $zv = u$ and let $\phi_y : R_u \to R_v$ and $\phi_z : R_v \to R_u$ be the respective Green's isomorphisms induced by y and z. There exists a right congruence $\eta \subseteq |^M_{R_v}$ so that there is a word graph isomorphism $\theta : R_u/\zeta \to R_v/\eta$.*

PROOF: By Theorem 4.2.11 part (i) we know that ϕ_y induces a word graph isomorphism from $S\Gamma(u)$ to $S\Gamma(v)$.

Define η as being the relation

$$\{(yu_1m, yu_2m) | (u_1, u_2) \in \zeta \cap (R_u \times R_u), m \in M\}$$

- η is reflexive as $(yu_1m)\eta = (yu_1m)\eta$ because $u_1\zeta = u_1\zeta$ for each $u_1 \in R_u$.
- η is symmetric because if $(yu_1m)\eta = (yu_2m)\eta$ then $u_1\zeta = u_2\zeta$ and so $u_2\zeta = u_1\zeta$ and so $(yu_2m)\eta = (yu_1m)\eta$.
- η is transitive because if $(yu_1m)\eta = (yu_2m)\eta$ and $(yu_2m)\eta = (yu_3m)\eta$ then $u_1\zeta = u_2\zeta$ and $u_2\zeta = u_3\zeta$ and so $u_1\zeta = u_3\zeta$ and so $(yu_1m)\eta = (yu_3m)\zeta$.
- η is consistent with multiplication on the right because if $(yu_1m)\eta = (yu_2m)\eta$ then $(yu_1mn)\eta = (yu_2mn)\eta$ for all $n \in M$.
- Suppose $(yu_1m)\eta = (yu_2m)\eta$. If $yu_1m \in R_v$ then $zyu_1m = u_1m \in R_u$ and remembering $\zeta \subseteq |^M_{R_u}$, if $u_2m \in R_u$ then $yu_2m \in R_v$. Conversely suppose that $yu_1m \in M \setminus R_v$ then by way of contradiction suppose that $yu_2m \in R_v$ then by symmetry we know that $yu_1m \in R_v$ and hence $yu_2m \in R_v \setminus R_u$. Therefore $\eta \subseteq |^M_{R_v}$.

Therefore η is a right congruence on M such that $\eta \subseteq |_{R_v}^M$ and so η is a right congruence on $S\Gamma(v)$.

Define $\theta : R_u/\zeta \to R_v/\eta$ by $(u_1\zeta)\theta = (yu_1)\eta$ for $u_1 \in R_u$.

- θ is a word graph homomorphism. This is because if $(u_1\zeta, x, u_2\zeta) \in E(S\Gamma(u)/\zeta)$ and we know that $(yu_1\eta)((x\tau)\eta) = (yu_1(x\tau))\eta$ because η is consistent with right multiplication, then $(u_1\zeta, x, u_2\zeta)\theta = (yu_1\eta, x, yu_2\eta)$ as required.

- θ is an injection. If $u_1\zeta, u_2\zeta \in R_u/\zeta$ such that $u_1\zeta \neq u_2\zeta$ then $yu_1 \neq yu_2$ by Lemma 4.2.10. Now suppose there existed $m \in M$ such that $(yu_1m)\eta = (yu_2m)\eta \in R_v/\eta$, then by the inverse property of $S\Gamma(v)/\eta$, $(yu)\eta = (yu_1mm^{-1})\eta = (yu_2mm^{-1})\eta = (yu_2)\eta$. Therefore $(u_1\zeta)\theta \neq (u_2\zeta)\theta$ as required.

- θ is a surjection. Suppose $v_1\eta \in R_v/\eta$ we know that $zv_1 \in R_u$ and so $(zv_1)\zeta \in R_u/\zeta$. Now $((zv_1)\zeta)\theta = (yzv_1)\eta = v_1\eta$ by Lemma 4.2.10 as required.

■

It now becomes apparent the extent of the limitations we must place on our discussion of right congruences on Schützenberger graphs. Given two different $\mathcal{R}$-classes R_u and R_v which are $\mathcal{L}$ related to each other then given a congruence $\zeta \subseteq |_{R_u}^M$ it is still necessary to find $\eta \subseteq |_{R_v}^M$ such that $S\Gamma(v)/\eta$ is isomorphic to $S\Gamma(u)/\zeta$. We shall see in the next chapter that the enumeration process bases itself on enumerating different $\mathcal{R}$-classes quite independently from each other.

The following theorem follows from Theorem 4.2.11 and Lemma 4.2.12.

Theorem 4.2.13. *Let $u, v \in M$ and let $e = uu^{-1}$ and $f = vv^{-1}$. Let $\zeta \subseteq |_{R_u}^M$ and $\eta \subseteq |_{R_v}^M$ be right congruences on M so that*

$$\eta = \{(yu_1m, yu_2m) | (u_1, u_2) \in \zeta \cap (R_u \times R_u),\ m \in M\}$$

for some $y \in M$. The following statements hold:

(i) *If $u\mathcal{D}v$ then there exists a word graph isomorphism*

$$\phi : S\Gamma(u)/\zeta \to S\Gamma(v)/\eta.$$

(ii) *If $u\mathcal{R}v$ then there exists a word graph isomorphism*

$$\phi : S\Gamma(u)/\zeta \to S\Gamma(v)/\eta$$

with the condition that $(e\zeta)\phi = f\eta$.

(iii) *If $u\mathcal{L}v$ then there exists a word graph isomorphism with*

$$\phi : S\Gamma(u)/\zeta \to S\Gamma(v)/\eta$$

the condition that $(u\zeta)\phi = v\eta$.

(iv) *If $u\mathcal{H}v$ then there exist word graph isomorphisms*

$$\psi, \phi : S\Gamma(u)/\zeta \to S\Gamma(v)/\eta$$

with the condition that $(e\zeta)\phi = f\eta$ *and* $(u\zeta)\phi = v\eta$.

(v) *If $u\zeta v$ (or equivalently if $u\eta v$) then there exists a word graph isomorphism*

$$\phi : S\Gamma(u)/\zeta \to S\Gamma(v)/\zeta$$

with the conditions that $(e\zeta)\phi = f\zeta$ *and* $(u\zeta)\phi = v\zeta$.

4.3 Graph Productions

Given $w = x_1 x_2 ... x_n$ $(x_i \in X \cup X^{-1})$ we need to be able to construct $S\Gamma(w)$. This process is similar but more difficult than the construction of word trees found in **Section 3.2**. Firstly we start with the *linear graph* of w. This is the birooted inverse word graph $(\alpha_w, \Gamma_w, \beta_w)$ where

$$V(\Gamma_w) = \{\alpha_w, \beta_w, \gamma_1, \gamma_2, ..., \gamma_{n-1}\}$$

and

$$\begin{aligned} E(\Gamma_w) = \ & \{(\alpha_w, x_1, \gamma_1), (\gamma_1, x_1^{-1}, \alpha_w), (\gamma_{n-1}, x_n, \beta_w), (\beta_w, x_n^{-1}, \gamma_{n-1})\} \\ & \cup \{(\gamma_{i-1}, x_i, \gamma_i), (\gamma_i, x_i^{-1}, \gamma_{i-1}) | 2 \leq i \leq n-1\}. \end{aligned}$$

EXAMPLE: If $X = \{x, y\}$ and $w = xx^{-1}yx^{-1}$ then Γ_w is:

$$\to \alpha \to_x \gamma_1 \leftarrow_x \gamma_2 \to_y \gamma_3 \leftarrow_x \beta$$

To convert the linear graph into the minimal graph Stephen introduces two constructions.

Determinations. Let (α, Γ, β) be a birooted inverse word graph over $X \cup X^{-1}$. This is a process of forcing Γ to be deterministic at a vertex γ.

Suppose we have $(\gamma, y, \delta_1), (\gamma, y, \delta_2) \in E(\Gamma)$ with $\delta_1 \neq \delta_2$ and $y \in X \cup X^{-1}$. We form a new birooted inverse word graph by taking the quotient of (α, Γ, β) by the equivalence relation on $V(\Gamma)$ generated by $\{(\delta_1, \delta_2)\}$. We call a sequence of determinations a *determination sequence.*

Elementary $\mathcal{P}$-expansions Let (α, Γ, β) be a birooted inverse word graph over $X \cup X^{-1}$. This construction is specific to a presentation $P = \langle X | U \rangle$. If $(r, s) \in U$ and Γ has a (δ_1, δ_2)-walk labelled by r but no (δ_1, δ_2)-walk labelled by s then we obtain a new birooted inverse word graph (α, Γ', β) by adjoining the linear graph of s to (α, Γ, β). Here we identify the start and end of $(\alpha_s, \Gamma_s, \beta_s)$ with δ_1 and δ_2 respectively.

If we have a right congruence ζ which is the intersection of all right congruences which contain the relation $V \subseteq (X \cup X^{-1})^* \times (X \cup X^{-1})^*$ then for $(r, s) \in V$ we have a *restricted elementary $\mathcal{P}$-expansion* such that if there is an (α, γ)-walk (note that α is the start) labelled by r but no (α, γ)-walk labelled by s then the new birooted inverse word graph is obtained by adjoining Γ_s at the start to α and at the end at γ. We need to restrict the application for right congruences because given an inverse monoid with $u \in M$ so that $ur \in R_u$ and $us \in R_u$, the only element $v \in R_u$ for which we can say *a priori* $(vr)\zeta = (vs)\zeta$ is $v = uu^{-1}$. This is because we know that $uu^{-1}rr^{-1} = rr^{-1}uu^{-1}$ and $uu^{-1}ss^{-1} = ss^{-1}uu^{-1}$ and so $uu^{-1}r$ is a right multiple of r and $uu^{-1}s$ is a right multiple of s. Indeed restricted elementary $\mathcal{P}$-expansions are sufficient for the same reason that in Todd-Coxeter coset enumeration of right congruence classes it is sufficient to check the application of a right congruence only on the first coset (see **Section 1.4**).

Restricted elementary $\mathcal{P}$-expansions are identical in character to elementary $\mathcal{P}$-expansions with the only difference that the former must start on the vertex uu^{-1}.

If A' is a determination of A it is a homomorphic image of A and so by Lemma 4.1.8 $L[A] \subseteq L[A']$. If $A' = (\alpha', \Gamma', \beta')$ is an elementary $\mathcal{P}$-expansion of $A = (\alpha, \Gamma, \beta)$ then Γ is a subgraph of Γ' and so $L[A] \subseteq L[A']$.

We will call an equivalence relation η on the vertices of a birooted inverse word graph (α, Γ, β) a *determinising equivalence* if $(\alpha\eta, \Gamma/\eta, \beta\eta)$ is deterministic. We need to show that the application of determinations is *confluent*.

Lemma 4.3.1 (Stephen). *If* $(\alpha_1, \Gamma_1, \beta_1)$ *and* $(\alpha_2, \Gamma_2, \beta_2)$ *are obtained from* $(\alpha_0, \Gamma_0, \beta_0)$ *by determination sequences, then there exists* $(\alpha_3, \Gamma_3, \beta_3)$ *which can be obtained from both* $(\alpha_1, \Gamma_1, \beta_1)$ *and* $(\alpha_2, \Gamma_2, \beta_2)$ *by determination sequences.*

The proof for this lemma is a strong induction on the sum of the number of determinations required to obtain $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ from $(\alpha_0, \Gamma_0, \beta_0)$ and can be found in Stephen's paper. Furthermore we have the following lemma which refers to birooted inverse word graphs in which it is impossible to make any further determinations.

Lemma 4.3.2 (Stephen). *A completely determinised birooted inverse word graph is a unique deterministic birooted inverse word graph.*

PROOF: As the set of equivalence relations on a given set is a complete lattice, the completely determinised graph of a given birooted inverse word graph is a well defined birooted inverse word graph. It is only necessary to show that the completely determinised graph of a given birooted inverse word graph is deterministic.

Let $A = (\alpha, \Gamma, \beta)$ be a birooted inverse word graph and let $A/\eta = (\alpha\eta, \Gamma/\eta, \beta\eta)$ be its completely determinised graph. Let $(\gamma\eta, y, \delta_1\eta), (\gamma\eta, y, \delta_2\eta) \in E(\Gamma/\eta)$. Let η_1 be any determinising quotient of A. Note that $\eta \subseteq \eta_1$ and also note that $\delta_1\eta_1 = \delta_2\eta_1$ since A/η_1 is deterministic. Now since η_1 is arbitrary, we see that $(\delta_1, \delta_2) \in \eta$, since δ_1 and δ_2 must be related by any determinising quotient of A. ■

It is useful to be able to characterise the determinising equivalence relation on a given birooted inverse word graph.

It is worth noting that the following results are valid for infinite graphs. Using the notation from **Section 3.2** where given a (γ_1, γ_2)-walk, P then $w(P)$ is the word labelling that walk.

Theorem 4.3.3 (Stephen). *Let* $A = (\alpha, \Gamma, \beta)$ *be a birooted inverse word graph, and let* η *be the largest determinising quotient of* A*. For* $\gamma_1, \gamma_2 \in V(\Gamma)$, $\gamma_1\eta = \gamma_2\eta$ *if and only if there is a* (γ_1, γ_2)*-walk,* P *in* Γ*, such that* $w(P)$ *is freely reducible, in the free group sense, to* ϵ*. That is* $w(P)$ *is an idempotent in* $\mathbf{F}_{\mathcal{IM}}(X)$.

PROOF: Let η be the least determinising quotient on $A = (\alpha, \Gamma, \beta)$, and let η_1 be the quotient on A defined by $\gamma_1\eta_1\gamma_2$ if and only if there exists a path from γ_1 to γ_2 which is labelled by a word which is freely reducible, in the free group sense to ϵ.

It is easily established that $\eta_1 \subseteq \eta$, by applying the definition of determination and employing a straightforward induction on the length of the word involved.

To complete the proof, we need only establish that Γ/η_1 is deterministic. Suppose that $(\delta\eta_1, y, \gamma_1\eta_1), (\delta\eta_1, y, \gamma_2\eta_1) \in E(\Gamma)$, then in Γ there exists (δ_1, y, γ_2'), such that $\delta_1\eta_1 = \delta_2\eta_1 = \delta\eta_1$. Now by definition of η_1 there exists paths $P_1 = (\gamma_1, \gamma_1'), P_2 = (\delta_1, \delta_2), P_3 = (\gamma_2', \gamma_2)$, such that $w(P_1), w(P_2)$ and $w(P_3)$ are freely reducible, in the free group sense, to ϵ. It then follows that $w(Q)$, where Q is the (γ_1, γ_2)-walk $Q = P_1(\gamma_1', y^{-1}, \delta_1)P_2(\delta_2, y, \gamma_2')P_3$, is freely reducible in the free group sense to ϵ and thus $\gamma_1\eta_1 = \gamma_2\eta_1$. hence A/η_1 is deterministic. ■

Next we have a *local confluence lemma* which shows the local confluence of determinations and elementary $\mathcal{P}$-expansions.

Lemma 4.3.4 (Stephen). *If $(\alpha_1, \Gamma_1, \beta_1)$ is obtained from $(\alpha_0, \Gamma_0, \beta_0)$ by either a determination or an elementary $\mathcal{P}$-expansion, and $(\alpha_2, \Gamma_2, \beta_2)$ is likewise obtained from $(\alpha_0, \Gamma_0, \beta_0)$ by a determination or an elementary $\mathcal{P}$-expansion, then there exists $(\alpha_3, \Gamma_3, \beta_3)$ which can be obtained from both $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ by sequences of determinations and elementary $\mathcal{P}$-expansions. Moreover at most one elementary $\mathcal{P}$-expansion is required in each sequence which derives $(\alpha_3, \Gamma_3, \beta_3)$.*

PROOF: There are four possible cases:

(i) $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ are both obtained by determinations of $(\alpha_0, \Gamma_0, \beta_0)$, in which case this is the same as Lemma 4.3.1.

(ii) $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ are both obtained by elementary $\mathcal{P}$-expansions of $(\alpha_0, \Gamma_0, \beta_0)$.

Suppose $(\alpha_1, \Gamma_1, \beta_1)$ is obtained by sewing on the (γ_1, γ_2) walk P_1 labelled by $x_1x_2...x_n$ and $(\alpha_2, \Gamma_2, \beta_2)$ is obtained by sewing the (γ_3, γ_4)-walk P_2 labelled by $y_1y_2...y_m$. Then we have the following sub cases.

(a) $x_1x_2...x_n$ does not label a (γ_1, γ_2)-walk in Γ_2, nor does $y_1y_2...y_m$ label a (γ_3, γ_4) walk in Γ_1.

In this case, the results of sewing the (γ_1, γ_2)-walk labelled by $x_1x_2...x_n$ onto Γ_2 and the result of sewing the (γ_3, γ_4)-walk labelled by $y_1y_2...y_m$ onto Γ_1 clearly lead to the same birooted inverse word graph, $(\alpha_3, \Gamma_3, \beta_3)$.

(b) $x_1x_2...x_n$ does not label a (γ_1, γ_2)-walk in Γ_2, but $y_1y_2...y_m$ does label a (γ_3, γ_4)-walk in Γ_1.

In this case sew the (γ_1, γ_2)-walk labelled by $x_1 x_2 ... x_n$ onto Γ_2. Call this path P. This yields a new birooted inverse word graph $(\alpha_2', \Gamma_2', \beta_2')$, which is the same birooted inverse word graph that would be obtained if we sewed the (γ_3, γ_4)-walk labelled by $y_1 y_2 ... y_m$ onto Γ_1. However, in Γ_1 we have a (γ_3, γ_4)-walk labelled by$y_1 y_2 ... y_m$. Call this path P'. In Γ_2 we have the path P that we sewed on and also P'. By a sequence of determinations, we can identify the path P with P'. the resulting birooted inverse word graph $(\alpha_3, \Gamma_4, \beta_3)$ is equal to $(\alpha_1, \Gamma_1, \beta_1)$.

(c) The symmetric opposite of (b), which is clearly equivalent.

(d) Both $x_1 x_2 ... x_n$ labels a (γ_1, γ_2)-walk in Γ_2 and $y_1 y_2 ... y_m$ labels a (γ_3, γ_4)-walk in Γ_1.

Let $(\alpha_3', \Gamma_3', \beta_3')$ be $(\alpha_0, \Gamma_0, \beta_0)$ with both the (γ_1, γ_2)-walk and the (γ_3, γ_4)-walk sewn on. Note that each of $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ can be obtained from $(\alpha_3', \Gamma_3', \beta_3')$ by a sequence of determinations. Thus by Lemma 4.3.1 there are sequences of determinations of both $(\alpha_1, \Gamma_1, \beta_1)$ and $(\alpha_2, \Gamma_2, \beta_2)$ which yield the same birooted inverse word graph $(\alpha_3, \Gamma_3, \beta_3)$.

(iii) $(\alpha_1, \Gamma_1, \beta_1)$ is obtained by an elementary $\mathcal{P}$-expansion from $(\alpha_0, \Gamma_0, \beta_0)$ and $(\alpha_2, \Gamma_2, \beta_2)$ is obtained by a determination by a determination of $(\alpha_0, \Gamma_0, \beta_0)$.

Suppose that $(\alpha_1, \Gamma_1, \beta_1)$ is obtained from $(\alpha_0, \Gamma_0, \beta_0)$ by sewing on the (γ_1, γ_2)-walk labelled by $x_1 x_2 ... x_n$ and $(\alpha_2, \Gamma_2, \beta_2)$ is obtained from $(\alpha_0, \Gamma_0, \beta_0)$ by the quotient generated by $\{(\gamma_3, \gamma_4)\}$. There are two sub cases to consider:

(a) If $(\alpha_2, \Gamma_2, \beta_2)$ does not have a (γ_1, γ_2)-walk labelled $x_1 x_2 ... x_n$, then we sew this path onto $(\alpha_2, \Gamma_2, \beta_2)$ and take the quotient of $(\alpha_1, \Gamma_1, \beta_1)$ induced by $\{(\gamma_3, \gamma_4)\}$. The resulting birooted word graphs are the same.

(b) If $(\alpha_2, \Gamma_2, \beta_2)$ does have a (γ_1, γ_2)-walk labelled $x_1 x_2 ... x_n$, then call this path P'. Now note that if we let $(\alpha_1', \Gamma_1', \beta_1')$ be the quotient of $(\alpha_1, \Gamma_1, \beta_1)$ induced by $\{(\gamma_3, \gamma_4)\}$, then $(\alpha_1', \Gamma_1', \beta_1')$ has two (γ_1, γ_2)-walks labelled by $x_1 x_2 ... x_n$. The one we originally sewed on and the one corresponding to P'. Identify these two paths by a sequence of determinations. The resulting birooted inverse word graph is $(\alpha_2, \Gamma_2, \beta_2)$.

(iv) The symmetric case opposite of (iii) which is clearly equivalent.

So in each case the lemma holds. ■

We may now talk about more general constructions.

$\mathcal{P}$-expansion. For a birooted inverse word graph (α, Γ, β): if $(\alpha_1, \Gamma_1, \beta_1)$ is obtained from (α, Γ, β) by an elementary $\mathcal{P}$-expansion, and $(\alpha_2, \Gamma_2, \beta_2)$ is the completely determinised graph of $(\alpha_1, \Gamma_1, \beta_1)$, then we say that $(\alpha_2, \Gamma_2, \beta_2)$ is obtained from (α, Γ, β) by a *$\mathcal{P}$-expansion.* We denote this by $(\alpha, \Gamma, \beta) \Rightarrow (\alpha_2, \Gamma_2, \beta_2)$. If $(\alpha_n, \Gamma_n, \beta_n)$ is obtained from (α, Γ, β) by a sequence of $\mathcal{P}$-expansions then we denote this by $(\alpha, \Gamma, \beta) \Rightarrow^* (\alpha_n, \Gamma_n, \beta_n)$.

It is interesting to compare this with Munn's method for constructing word trees from a word in $\mathbf{F}_{\mathcal{IM}}(X)$ (see **Section 3.2**). Although Munn constructed the trees vertex by vertex it would be just as valid to start with a linear graph and simply determinise it. Also note that in $\mathbf{F}_{\mathcal{IM}}(X)$ there are no relations and so elementary $\mathcal{P}$-expansions can never be applied.

4.4 Approximations to Schützenberger Graphs

Definition 4.4.1. For $u \in (X \cup X)^*$, *an approximate graph of* $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ is a birooted inverse word graph $A = (\alpha, \Gamma, \beta)$ with the properties $u \in L[A]$ and $w\tau \geq u\tau$ for all $w \in L[A]$.

An approximate graph is (usually) a non-deterministic automaton, A, which shares the important property of $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ that $L[A]\tau \subseteq L[(uu^{-1}\tau, S\Gamma(u\tau), u\tau)]\tau$. In particular the linear graph of u is an *approximation* to $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$.

Lemma 4.4.2 (Stephen). *If $A = (\alpha, \Gamma, \beta)$ is a deterministic birooted inverse word graph and $u \in L[A]$, then for every $v \in (X \cup X^{-1})^*$ with $v\rho \geq u\rho$ then $v \in L[A]$.*

PROOF: Let $\mathcal{I}(V(\Gamma))$ be the symmetric inverse monoid on $V(\Gamma)$. We define a natural action of $(X \cup X^{-1})^*$ on $\mathcal{I}(V(\Gamma))$. We define a homomorphism ϕ : $(X \cup X^{-1})^* \to \mathcal{I}(V(\Gamma))$ by $w\phi = \psi_w$ where $\psi_w : V(\Gamma) \to V(\Gamma)$ is defined by $\gamma\psi = \gamma w$. Now notice that each ψ_w is a one-to-one map on $V(\Gamma)$, since Γ is deterministic, and that $\psi_{w^{-1}}$ is an inverse of ψ_w since Γ is an inverse word graph. Moreover, since the maps are defined by right multiplication it is easy to see that ϕ is a homomorphism.

Let v be such that $v\rho \geq u\rho$. Note that by the universal property of $(X \cup X^{-1})^*/\rho$, $v\phi \geq u\phi$, so that $(uu^{-1}v)\phi = u\phi$, ie., $\psi_{uu^{-1}v} = \psi_u$. In particular, $\alpha\psi_{uu^{-1}v} = \alpha\psi_u = \beta$. Now since $\alpha\psi_{uu^{-1}} = \alpha$, it is clear that $\alpha\psi_v = \beta$, that is, $v \in L[A]$ as required. ■

The following lemmas and their corollaries demonstrate that throughout the process of finding the minimal automaton recognising the word u, the automata are always approximations. Stephen's proofs are omitted because they are quite technical.

Lemma 4.4.3 (Stephen). *Suppose the word graph (α, Γ, β) is an approximate graph of $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$. If $(\alpha', \Gamma', \beta')$ is a determination of (α, Γ, β), then $(\alpha', \Gamma', \beta')$ is an approximate graph of $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$.*

Corollary 4.4.4. *Let $\zeta \subseteq |^M_{R_u}$ be a right congruence on M. Let (α, Γ, β) be an approximate graph of $((uu^{-1}\tau)\zeta, S\Gamma(u\tau)/\zeta, (u\tau)\zeta)$. If $(\alpha', \Gamma', \beta')$ is a determination of (α, Γ, β), then $(\alpha', \Gamma', \beta')$ is an approximate graph of $((uu^{-1}\tau)\zeta, S\Gamma(u\tau)/\zeta, (u\tau)\zeta)$.*

PROOF: This follows from Lemma 4.4.3 and the fact that $S\Gamma(u\tau)/\zeta$ is deterministic. ■

Lemma 4.4.5 (Stephen). *Suppose the word graph (α, Γ, β) is an approximate graph of $((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$. If $(\alpha', \Gamma', \beta')$ is obtained from (α, Γ, β) by an elementary $\mathcal{P}$-expansion, then $(\alpha', \Gamma', \beta')$ is an approximate graph of $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$.*

Corollary 4.4.6. *Let $\zeta \subseteq |^M_{R_u}$ be a right congruence on M. Let (α, Γ, β) be an approximate graph of $((uu^{-1}\tau)\zeta, S\Gamma(u\tau)/\zeta, (u\tau)\zeta)$. If $(\alpha', \Gamma', \beta')$ is obtained from (α, Γ, β) by a restricted elementary $\mathcal{P}$-expansion, then $(\alpha', \Gamma', \beta')$ is an approximate graph of $((uu^{-1}\tau)\zeta, S\Gamma(u\tau)/\zeta, (u\tau)\zeta)$.*

PROOF: This follows from Lemma 4.4.5 and the fact that restricted elementary $\mathcal{P}$-expansions always append to the start and $(uu^{-1}r)\tau = (uu^{-1}rr^{-1}r)\tau = (r(r^{-1}uu^{-1}r))\tau$ is a right multiple of $r\tau$. ■

Theorem 4.4.7. *Let $u, v \in (X \cup X^{-1})^*$ and let $A = (\alpha, \Gamma, \beta)$ be an approximate graph of $((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$. If $v\tau \geq u\tau$, then there exists a sequence of $\mathcal{P}$-expansions $(\alpha, \Gamma, \beta) \Rightarrow^* (\alpha'\Gamma', \beta')$ such that $v \in L[(\alpha', \Gamma', \beta')]$.*

PROOF: We will assume, without loss of generality, that (α, Γ, β) is deterministic. Moreover, without loss of generality, we will assume that $u\tau = v\tau$ since $uu^{-1}v\tau = u\tau$ and $uu^{-1}v \in L[(\alpha'\Gamma', \beta')]$ if and only if $v \in L[(\alpha', \Gamma', \beta')]$.

Note that every elementary τ transition is either an elementary ρ transition or an elementary U transition. We therefore have two cases.

1. If $u\rho = v\rho$ then $v \in L[(\alpha, \Gamma, \beta)]$ by Lemma 4.4.2.
2. Suppose there is (r, s) (or (s, r)) in U such that $v = u_1 r u_2$ and $u = u_1 s u_2$. Now $u_1 s u_2$ labels an (α, β)-walk in (α, Γ, β), where the subpath labelled by s is a (γ_1, γ_2)-walk. If r labels a (γ_1, γ_2)-walk then $v \in L[(\alpha, \Gamma, \beta)]$, in this case let $(\alpha', \Gamma', \beta') = (\alpha, \Gamma, \beta)$. Otherwise sew on the (γ_1, γ_2)-walk labelled by r to get $(\alpha_1, \Gamma_1, \beta_1)$ and let $(\alpha', \Gamma', \beta')$ be its determinised form. Now note that in each case $v \in L[(\alpha', \Gamma', \beta')]$.

■

Corollary 4.4.8. *Let $u, v \in (X \cup X^{-1})^*$, let $\zeta \subseteq |^M_{R_u}$ be a right congruence on M and let $A = (\alpha, \Gamma, \beta)$ be an approximate graph of $((uu^{-1}\tau)\zeta, S\Gamma(u\tau)/\zeta, (u\tau)\zeta)$. If $v\tau \geq u\tau$, then there exists a sequence of $\mathcal{P}$-expansions $(\alpha, \Gamma, \beta) \Rightarrow^* (\alpha'\Gamma', \beta')$ such that $v \in L[(\alpha', \Gamma', \beta')]$.*

PROOF: This follows from Theorem 4.4.7 and Corollary 4.2.8. ■

Definition 4.4.9. A *closed* approximate graph is one in which no non-trivial $\mathcal{P}$-expansions or determinations can be carried out.

The next theorem is central to solving the word problem.

Theorem 4.4.10 (Stephen). *Let $w \in (X \cup X^{-1})^*$ and let (α, Γ, β) be an approximate graph of $(ww^{-1}\tau, S\Gamma(w\tau), w\tau)$. If $(\alpha, \Gamma, \beta) \Rightarrow^* (\alpha', \Gamma', \beta')$ where $(\alpha', \Gamma', \beta')$ is closed, then $(\alpha', \Gamma', \beta')$ is isomorphic to the Schützenberger graph, $(ww^{-1}\tau, S\Gamma(w\tau), w\tau)$.*

PROOF: This follows from Lemma 4.1.11 and Theorem 4.4.7 ■

Theorem 4.4.10 combined with Theorem 4.2.11 (v) demonstrates a method for solving the word problem in an inverse monoid M. Similarly when Theorem 4.4.10 is combined with Theorem 4.2.11 (i) to (iv) we have a method for finding whether two elements of M are $\mathcal{D}$, $\mathcal{R}$, $\mathcal{L}$ or $\mathcal{H}$ related.

We shall have to generalise Theorem 4.4.10 for right congruences. This is no problem because right quotients of Schützenberger graphs are trim, inverse and deterministic. Formally we have the following theorem.

Theorem 4.4.11. *Let* $\zeta \subseteq |^M_{R_u}$ *be a right congruence. Let* $w \in (X \cup X^{-1})^*$ *and let* (α, Γ, β) *be an approximate graph of* $((ww^{-1}\tau)\zeta, S\Gamma(w\tau)/\zeta, (w\tau)\zeta))$. *If* $(\alpha, \Gamma, \beta) \Rightarrow^* (\alpha', \Gamma', \beta')$ *where* $(\alpha', \Gamma', \beta')$ *is closed, then* $(\alpha', \Gamma', \beta')$ *is isomorphic to the Schützenberger graph,* $((ww^{-1}\tau)\zeta, S\Gamma(w\tau)/\zeta, (w\tau)\zeta)$.

4.5 Comments on Solving the Word Problem

Given $u, v \in (X \cup X^{-1})^*$, the word problem is thus solved in the following two steps:

1. Find the minimal automata $A_u = (uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ and $A_v = (vv^{-1}\tau, S\Gamma(v\tau), v\tau)$ recognising u and v respectively.
2. $u = v$ in M if and only if u labels a $(vv^{-1}\tau, v\tau)$-walk in A_v and v labels a $(uu^{-1}\tau, u\tau)$-walk in A_u.

Theorem 4.5.1. *The second step is correct.*

PROOF: If $u\tau = v\tau$ then we know that $u \in L[(vv^{-1}\tau, S\Gamma(v\tau), v\tau)]$ and similarly $v \in L[(uu^{-1}\tau, S\Gamma(u\tau), u\tau)]$ by Theorem 4.4.7.

Conversely if $u \in L[(vv^{-1}\tau, S\Gamma(v\tau), v\tau)]$ and $v \in L[(uu^{-1}\tau, S\Gamma(u\tau), u\tau)]$ then $(vv^{-1}u)\tau = u\tau$ and $(uu^{-1}v)\tau = v\tau$ and so

$$(uu^{-1})\tau = (vv^{-1}uu^{-1}vv^{-1})\tau = (uu^{-1}vv^{-1})\tau$$

and similarly $(vv^{-1})\tau = (uu^{-1}vv^{-1})\tau$ and so $(uu^{-1})\tau = (vv^{-1})\tau$ hence $R_{u\tau} = R_{v\tau}$. Hence u labels a $(vv^{-1}\tau, v\tau)$-walk in $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ and v labels a $(uu^{-1}\tau, u\tau)$-walk in $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ and so

$$u\tau = (uu^{-1}u)\tau = (uu^{-1}v)\tau = (vv^{-1}v)\tau = v\tau.$$

■

We can see that the word problem is solvable if there is a finite way of generating R_u and R_v. Usually this requires both R_u and R_v to be finite. Of course, it is possible that R_u and R_v are both finite while M is infinite, and so Stephen's technique for solving the word problem is indeed quite powerful. He manages to prove the following quite remarkable result.

Theorem 4.5.2 (Stephen). *Let M be the inverse monoid given by the presentation $\langle X | {w_i}^{n_i} = {w_i}^{n_i+k_i}, i \in I\rangle$ where I is some finite index set. If $0 < k_i \leq n_i$ for all $i \in I$, then M has decidable word problem. In particular, $(uu^{-1}\tau, S\Gamma(u\tau), u\tau)$ is finite and effectively constructible for all $u \in (X \cup X^{-1})^*$.*

In particular if $n_i = k_i = 1$ for each $i \in I$ then the above presentation becomes $\langle X | w_i = {w_i}^2\rangle$, which considered as a group presentation, is equivalent to $\langle X | w_i = \epsilon\rangle$ after free cancelation - ie. any finite group presentation! Of course if we consider the group presentation $P = \langle X | w_i = \epsilon\rangle$ as an inverse monoid presentation then there is no guarantee that the word problem is decidable for the inverse monoid presented by P.

Chapter 5

Inverse Monoid Enumerator

Theorems 4.2.11 and 4.4.10 generalise the results found in **Section 3.2**. We are now suitably equipped to enumerate $\mathcal{R}$-classes.

Throughout this chapter I will be talking about an inverse monoid $M = (X \cup X^{-1})^*/\tau$ finitely presented as an inverse monoid by $\langle X | U \rangle$.

Everything in this chapter is new and constitute the main results of the thesis.

5.1 $\mathcal{R}$-class Enumerator

A word graph Γ can be considered as an incomplete coset table. This is done by simply taking the vertices as rows, the columns as the labelling set and the entry $T(c, x)$ in row c and column x is taken as the the edge $(c, x, T(c, x))$.

I shall now proceed to describe the algorithm.

5.1.1 The Data Structures

- The immutable *presentation for* M, $\langle X | U \rangle$ stored as a list of generators and their inverses and a list of pairs of words.
- The *set of cosets* is the mutable set C which is a set of positive integers. Initially $C := \{1\}$.

- The *coset table* T is mutable. With columns labelled by $X \cup X^{-1}$ and rows labelled by C with entries from the set $C \cup \{\perp\}$ where $\perp$ is the *empty symbol*. It starts with an empty row of $\perp$'s labelled by 1.

- The *coincidence set* $K \subseteq C \times C$ is mutable and is considered as a stack.

- The function $r : C \to C \cup 0$ ($r(c)$ is always less then c) is for replacing deleted rows - if $r(c) = 0$ then c has not been deleted. Initially $r(1) := 0$.

5.1.2 The Subroutines

The full names of the procedures are given in bold while the part of the names in italics are their shorthand names. Some procedures simply change the data structure while others will return a value, others will do both.

Replace

- Parameter: $c \in C$

- Locals: None

- While $r(c) > 0$ then $c := r(c)$

- Return c

Create a *New* Definition

- Parameters: $c \in C$ and $x \in X \cup X^{-1}$.

- Local: d

Do the following in order:

- Add $d := \max(C) + 1$ to C.

- Add an empty row onto T labelled by d.

- Define $r(d) := 0$

- Define $cx := d$

- Define $dx^{-1} := c$
- Return d.

Trace a Relation

- Parameters: $c \in C$ and $(u = u_1u_2...u_k, v = v_1v_2...v_m) \in U$.
- Locals: i, s, d, e

Do the following in order:

- If $cu = \perp$ and $cv = \perp$ then do nothing.
- If $cu_1...u_i \neq \perp$ and $cu_1...u_{i+1} = \perp$ $(1 \leq i \leq k)$ and $cv = d \neq \perp$ define $s := cu_1...u_i$ then while $i \leq k$
  - Do $s := \mathbf{New}(s, u_i)$
  - Let $i := i + 1$.
- If $cv_1...v_i \neq \perp$ and $cv_1...v_{i+1} = \perp$ $(1 \leq i \leq m)$ and $cu = d \neq \perp$ define $s := cv_1...v_i$ then while $i \leq m$
  - Do $s := \mathbf{New}(s, v_i)$
  - Let $i := i + 1$.
- If $cu = d \neq \perp$ and $cv = e \neq \perp$ and $d \neq e$ then push (d, e) onto K.
- **Identify**

Identify Coincidences

- Parameters: None
- Locals: d, e, s, x

While K is not empty do the following:

- Pop (d, e) from K.

- Let $d :=$ **Replace**(d) and let $e :=$ **Replace**(e)
- If $d \neq e$ then (assuming with out loss of generality that $d < e$) do the following
  - For each entry equal to e in T replace e by d.
  - For each $x \in X \cup X^{-1}$ if $dx = \perp$ then replace dx by ex otherwise if $ex \neq \perp$ replace dx by $\min(dx, ex)$ and add (dx, ex) to K.
  - For each pair (s, e) or (e, s) in K replace with (s, d) and (d, s) respectively.
- Let $r(e) := d$ and mark row e as *complete*

5.1.3 The Main Procedure

The algorithm runs as follows.

To enumerate R_u first generate the coset table of the word tree of $u = x_1x_2...x_n$ as an element of $\mathbf{F}_{\mathcal{IM}}(X)$. To do this start with an empty coset table with only the 1st row with the variable $c := 1$. For each x_i where i runs from 1 to n do the following.

- if $cx_i = \perp$ then let $c :=$ **New**(c, x_i),
- else $c := cx_i$.

This preliminary procedure will set up the table before applying any relations. It is all that is needed to enumerate R_u in $\mathbf{F}_{\mathcal{IM}}(X)$ (remember that in this case R_u is always finite). Now resetting $c := 1$ we proceed as follows.

- Repeat
- For each $1 \leq d \leq c$ so that $r(d) = 0$ and for each $(u, v) \in U$ do the following
  - **Trace**$(d, (u, v))$
- Mark coset c as *complete*

- Let A be the set of *incomplete* cosets. If $A \neq \emptyset$ then let $c := \min(A)$.
- Until $A = \emptyset$
- **Tidy up** T
- Return T

The **Tidy up** process removes all rows which have been deleted (ie. $r(c) > 0$), it also renumbers the cosets so that the rows read $1, 2, 3, \ldots$ etc.

5.1.4 Proof that the $\mathcal{R}$-class Algorithm Enumerates R_u

It should be noted that there is no particular reason that this algorithm should terminate. It should also be remembered that there is no particular reason that R_u should be finite.

I shall talk about stages. The first stage, stage 1, starts just after the data structures have been set up. A new stage starts every time the For loop in the Main Procedure is started.

At the end of each stage, s, we call the coset table, T, (without rows whose r value is greater than 0) T_s. A map from coset tables to the corresponding word graphs is defined by $\theta : T_s \mapsto \Gamma_s$ where the cosets (whose r value is 0) map to vertices and edges are mapped to $(c\theta, x, d\theta)$ when $T_s(c, x) = d$ in T_s.

It is clear that at every stage T_s is a fully determinised word graph because each entry in the table is well defined.

A careful inspection of the **New**, **Trace** and **Identify** subroutines reveals the following three lemmas.

Lemma 5.1.1. *Γ_1 is the determinised linear graph of u that is Γ_1 is the birooted word tree (α_u, T_u, β_u).*

PROOF: Γ_1 is produced by the preliminary procedure which runs through each of the letters in u. Let us regard c as a pointer in the tree. The only way in which the table may induce a loop in the corresponding graph is that the pointer moves to a non-adjacent vertex on the graph. As at each step in the For loop, either a new vertex is created which is adjacent to the pointer or the pointer moves to an adjacent vertex then at no stage is a loop induced.

It is only necessary to notice that $P = (\alpha, \alpha x_1, \alpha x_1 x_2, ..., \alpha u)$ is a spanning walk on Γ_1 to complete the proof. ∎

Lemma 5.1.2. *At each stage, s, the operation of the subroutine* ***Trace***$(c, (u, v))$ *on the coset table* T_s *either:*

- *If neither side of* (u, v) *can be traced from c then the subroutine does nothing.*
- *If* u *can be traced from c then the subroutine adds a determinised linear graph of* v *to vertex c.*
- *If* v *can be traced from c then the subroutine adds a determinised linear graph of* u *to vertex c.*
- *If both* u *and* v *can be traced from c then it adds* (cu, cv) *to the coincidence set.*

PROOF: Clear. ∎

Lemma 5.1.3. *At each stage, s, the operation of the subroutine* ***Identify*** *on the coset table* T_s *corresponds, by the mapping* θ*, to a series of* $\mathcal{P}$*-expansions on* Γ_s*. Moreover given any coset c* ***Identify*** *does every possible* $\mathcal{P}$*-expansion starting at* c.

PROOF: For each $(d, e) \in K$, the $\mathcal{P}$-expansion carried out is attaching vertex d to vertex e. Making the table consistent after this corresponds, by the mapping θ, to a determination sequence. We know that a particular elementary $\mathcal{P}$-expansion corresponds to a particular relation (u, v) on vertex c because the only place (d, e) can be added to the stack is in the subroutine **Trace** when we discover that $d = cu$ and $e = cv$.

Identify does every possible $\mathcal{P}$-expansion because in Lemma 5.1.2 we see that Γ_s is expanded so as to include every path labelled by u starting at c when there is a path labelled by v starting at c where $(u, v) \in U$. ∎

Theorem 5.1.4. Γ_s *is an approximate graph of* $((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$ *at each stage s.*

PROOF: We know Γ_1 is an approximate graph of $((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$ by Lemma 5.1.1 and the fact that the linear graph of u is an approximate graph of $(uu^{-1}, S\Gamma(u), u)$. Hence by Lemma 4.4.5 and Lemma 5.1.3 $\Gamma_2, \Gamma_3, \ldots$ are approximate graphs of $((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$ by induction. ■

Theorem 5.1.5. *The algorithm terminates if and only if R_u is finite in which case there is a stage s such that $\Gamma_s = ((uu^{-1})\tau, S\Gamma(u\tau), u\tau)$. In this case the table T is returned where $T\theta = S\Gamma(u)$.*

PROOF: This follows from Theorem 4.4.10. ■

5.2 Examples

It will be useful to give a couple of demonstrations of the $\mathcal{R}$-class algorithm before proceeding.

EXAMPLE: Let M be presented by $\langle x | x^4 = x^2 \rangle$. Let us apply the $\mathcal{R}$-class algorithm to the $\mathcal{R}$-class generated by x. The coset table for the word tree of x is as follows:

Cosets	x	x^{-1}	$r(c)$
1	2	$\perp$	0
2	$\perp$	1	0

Whether starting at coset 1 or coset 2 we notice that it is impossible to trace either side of our relation through the table. Therefore the **Trace** procedure does nothing and the algorithm simply returns the above table.

In this case R_x is precisely the same as the $\mathcal{R}$-class generated by x in $\mathbf{F}_{\mathcal{IM}}(x)$ that is the elements x and xx^{-1}.

EXAMPLE: Let M be as in the previous example but this time apply the $\mathcal{R}$-class algorithm to R_{x^3}. The coset table for the linear graph is as follows:

Cosets	x	x^{-1}	$r(c)$
1	2	$\perp$	0
2	3	1	0
3	4	2	0
4	$\perp$	3	0

Starting at coset 1 it is possible to trace the right hand side of our relation but not the left hand side. The **Trace** procedure forces us to define a new coset 5 such that $4x = 5$ and then another coset 5 such that $4x = 5$ we have:

Cosets	x	x^{-1}	$r(c)$
1	2	$\perp$	0
2	3	1	0
3	4	2	0
4	5	3	0
5	$\perp$	4	0

But we also are forced to add $(5, 3)$ to K because $1x^2 = 3$ and $1x^4 = 5$. Now running procedure **Identify** we set $r(5) = 3$, we replace all occurrences of 5 by 3 in the table and we are forced to add $(2, 4)$ on the stack, we have:

Cosets	x	x^{-1}	$r(c)$
1	2	$\perp$	0
2	3	1	0
3	4	2	0
4	3	3	0
5	$\perp$	4	3

Continuing we set $r(4) = 2$ and replace all 4's by 2's in the table and we are forced to add $(1, 3)$ onto the stack. Finally we are left with:

Cosets	x	x^{-1}	$r(c)$
1	2	2	0
2	1	1	0
3	2	2	1
4	1	1	2
5	$\perp$	2	3

The algorithm then checks coset 1 again and finds it is consistent with the relation. It then checks coset 2 and finds that this is also consistent with the relation and then tidies the table so that it looks like:

Cosets	x	x^{-1}	$r(c)$
1	2	2	0
2	1	1	0

This $\mathcal{R}$-class is isomorphic to the group C_2 which is the group obtained from the presentation of M. Indeed it is not too difficult to see that M has 4 $\mathcal{R}$-classes:

1. The trivial $\mathcal{R}$-class generated by the identity and containing only the identity.
2. $R_x = R_{xx^{-1}}$ which is of order two.
3. $R_{x^{-1}} = R_{x^{-1}x}$ which is also of order two.
4. $R_{x^2} = R_w$ where w is any word in $(X \cup X^{-1})^*$ which has either x^2 or x^{-2} as a subword. This $\mathcal{R}$-class is of order 2 and we may use x^2 and x^3 as the canonical forms. R_{x^2} is the cyclic group of order two with x^2 being the identity. The difference between R_{x^2} and R_x is that R_x is not closed under multiplication, in particular $x * x \notin R_x$. Similarly $x^{-1} * x^{-1} \notin R_{x^{-1}}$. As regards the coset tables both R_x and $R_{x^{-1}}$ have holes where as R_{x^2} does not.

M therefore has 7 elements. The inverse semigroup given by the same presentation is the same except that it does not contain an identity element.

It is worth commenting at this point that the columns of a standard (group) Todd-Coxeter algorithm are permutations of the set of cosets by a generator. For the

$\mathcal{R}$-class enumeration algorithm we see that the columns have holes (or $\perp$'s) in them but are otherwise partial one-to-one mappings of the set of cosets in to itself. If we recall Theorem 2.4.5 (the Wagner representation theorem) then it should not be surprising that generators perform partial injections of elements (cosets) of $\mathcal{R}$-classes.

5.3 Generating the Set of $\mathcal{R}$-Classes

An inverse monoid is the disjoint union of $\mathcal{R}$-classes, therefore we can enumerate the entire monoid if we can systematically generate all the $\mathcal{R}$-classes of an inverse monoid M. I have already hinted how to do this in the last example. I use an orbit algorithm which I shall proceed to describe.

5.3.1 The Data Structures

The only data structure is a stack, R, of pairs. The first element of each pair is a word (a *representative word*) $u \in (X \cup X^{-1})^*$. The second element of each pair is the coset table for R_u denoted by T_u. Initially R starts with a single element (ϵ, T_ϵ) where T_ϵ is found using the $\mathcal{R}$-class algorithm on R_ϵ.

5.3.2 The Subroutines

***Enumerate* $\mathcal{R}$-class**

- Parameter: $w \in (X \cup X^{-1})^*$

This is the $\mathcal{R}$-class algorithm described in section 5.1

Check whether a *Path* can be Traced in a Table

- Parameters: $(u, T_u) \in R, v = v_1 v_2 ... v_n \in (X \cup X^{-1})^*$
- Locals: c, i
- Let $c := 1$

- For each v_i where i runs from 1 to n do the following:
  - $c := T_u(c, v_i)$
  - if $c = \perp$ then Return $False$
- Return $True$

5.3.3 The Orbit Algorithm

For each $(w, T_w) \in R$ and each $x \in X \cup X^{-1}$ do the following:

- Let $T_{xw} =$ **Enumerate**(R_{xw})
- Let $F := False$.
- For each $(u, T_u) \in R$ do the following while $F = False$
  - Let $F :=$ **Path**$((u, T_u), xw)$ and **Path**$((xw, T_{xw}), u)$.
- If $F = False$ then Push (xw, T_{xw}) onto R

Return R

NOTE: As $\mathcal{R}$-classes are added to R (they are never taken off), the outer For loop simply continues running through the new $\mathcal{R}$-classes. The algorithm will only terminate when new $\mathcal{R}$-classes are not being generated. If this never happens then M has an infinite number of (finite) $\mathcal{R}$-classes and it should not be expected that the algorithm terminate.

NOTE: This is a quite inefficient algorithm because it tends to repeatedly generate the same $\mathcal{R}$-classes.

5.3.4 Proof that the Orbit Algorithm generates M

Before proceeding let us examine exactly how the **Path** subroutine can be used to check the equality of $\mathcal{R}$-classes. We merely need to restate Corollary 4.2.7 (ii).

Lemma 5.3.1. *Assume we are given a monoid presented by $\langle X|U\rangle$ with $u, v \in (X \cup X^{-1})$ so that u generates a finite $\mathcal{R}$-class with table T_u. Then* ***Path***$((u, T_u), v) = True$ *if and only if $uu^{-1}v = v$.*

Suppose that u and v generate finite $\mathcal{R}$-classes in M. We know that if $uu^{-1}v = v$ and $vv^{-1}u = u$ then $uu^{-1} = uu^{-1}vv^{-1} = vv^{-1}$ and so if $\mathbf{Path}((u, T_u), v) = True$ and $\mathbf{Path}((v, T_v), u) = True$ then $R_u = R_v$. As the converse is clearly true this last lemma allows us to check $\mathcal{R}$-class equality.

Theorem 5.3.2. *Given a monoid M, the Orbit Algorithm will terminate if and only if M is finite. Upon termination the list of $\mathcal{R}$-classes of M with their representatives will be returned.*

PROOF:

The algorithm can only be expected to generate M if M is finite. Hence we assume that M has a finite number of finite $\mathcal{R}$-classes. I shall denote the set of $\mathcal{R}$-classes of M by $\mathbf{R}$. I need to show that at each stage of the Orbit Algorithm that each element, (u, T_u), of R corresponds to exactly one element, R_u, of $\mathbf{R}$ and that the algorithm terminates with a one-to-one corresponds from $\mathbf{R}$ to R. It is useful to think of an injective mapping from R to $\mathbf{R}$ which eventually becomes surjective as well. Hence there are two things to prove.

1. "one-to-one" If $(u, T_u) \in R$ and given $v \in (X \cup X^{-1})^*$ such that $u \neq v$ and $R_u = R_v$, then at any point in the calculation $(v, T_v) \notin R$.

2. "eventually onto" There is a total ordering ("*length-by-reverse-lexicographic*") $\leq$ on $(X \cup X^{-1})^*$ such that given any $u \in (X \cup X^{-1})^*$ then after a finite time there exists $v \leq u$ such that $R_u = R_v$ and $(v, T_v) \in R$.

1. "one-to-one"

   Suppose $(u, T_u) \in R$, $R_u = R_v$ and $v = xw$ where $x \in X \cup X^{-1}$ and $(w, T_w) \in R$. At some point the outer For loop will examine (w, T_w) and hence T_v :=**Enumerate**(R_v) is calculated. It should be noted that if $u = v$ in $(X \cup X^{-1})^*$ then this situation never arises as the Orbit Algorithm never checks the same word twice.

   The inner For loop examines each element in R including (u, T_u). Now $\mathbf{Path}((u, T_u), v) = True$ if and only if $uu^{-1}v = v$, but by assumption $v\mathcal{R}u\mathcal{R}uu^{-1}$ and so $\mathbf{Path}((u, T_u), v) = True$. Similarly $\mathbf{Path}((v, T_v), u) = True$. Hence the algorithm will not Push (v, T_v) onto R. We now only need to note that the algorithm only pushes pairs onto R immediately after they have been examined.

2. "eventually onto"

   I define *length-by-reverse-lexicographic* as follows. Suppose there is a total ordering $\leq$ on $X \cup X^{-1}$ so that the outer For loop in the Orbit Algorithm runs through each $x \in X \cup X^{-1}$ from least to greatest. We extend the ordering to $(X \cup X^{-1})^*$ by saying that if $|u| < |v|$ then $u < v$ while if $|u| = |v|$ and $u = u_j ... u_2 u_1 x_i ... x_2 x_1$, $v = v_k ... v_2 v_1 x_i .... x_2 x_1$ where $x_1, ..., x_i, u_1, ..., u_j, v_1, ..., v_k \in X \cup X^{-1}$ with $u_1 \neq v_1$ then $u < v$ if and only if $u_1 < v_1$. The Orbit Algorithm will always generate new pairs, (u, T_u), where u is greater than or equal to all previous words it has examined.

   Given $u \in (X \cup X^{-1})^*$, let $v := \min(w \in (X \cup X^{-1})^* | w \leq u, w\mathcal{R}u)$. Suppose $v = x_n ... x_2 x_1$ ($x_i \in (X \cup X^{-1})^*$). Let $v_i := x_i ... x_2 x_1$ for each $1 \leq i \leq n$ and let $v_0 = \epsilon$. The algorithm starts with $(v_0, T_{v_0}) \in R$. Suppose that $(v_k, T_{v_k}) \in R$ $(0 \leq k < n)$. We know that the algorithm will then at some point examine $R_{v_{k+1}}$. If $(v_{k+1}, T_{v_{k+1}})$ is not added to R then there is some $w < v_{k+1}$ such that $R_w = R_{v_{k+1}}$. However this means that $R_{x_n ... x_{k+3} x_{k+2} w} = R_{x_n ... x_{k+3} x_{k+2} v_{k+1}} = R_v$ with $x_n ... x_{k+3} x_{k+2} w < v$ which contradicts our assumption. Therefore the algorithm will generate $(v_1, T_{v_1}), (v_2, T_{v_2}), ..., (v_n, T_{v_n}) = v$ as required.

■

5.4 Comments and Improvements

The Orbit Algorithm has a remarkable strength as well as an important weakness. The strength lies in the ease in which it is possible to decide whether two $\mathcal{R}$-classes are identical. This strength is surprising because the problem of deciding whether two $\mathcal{R}$-classes are identical is similar to a special case of the *graph isomorphism problem* (which asks whether two graphs are isomorphic) because $\mathcal{R}$-classes are labelled directed graphs. There is no known polynomial time solution to the graph isomorphism problem. However the **Path** subroutine is polynomial and, indeed, very quick. To understand this, consider that every $\mathcal{R}$-class is not recorded as merely a graph but as a graph along with a generating word which acts as a "key". The first versions of the algorithm used a recursive algorithm to check whether the tables for two words were identical ie. it repeatedly solved special cases of the graph isomorphism problem.

The weakness, as I have pointed out, is that a single $\mathcal{R}$-class will be generated several times. There does not appear to be a way of solving this problem. The free inverse monoid algorithm developed in **Section 3.1** had a form of coset collapse which was dependent on the free inverse monoid identities rather than the relations and because of that it works more or less like a standard Todd-Coxeter. However when we introduce explicit relations, the free inverse monoid algorithm has to check both identities and relations and there is no way of immediately telling how they interact with one another. For this reason it seems doubtful that there is a Todd-Coxeter style algorithm which does not follow a similar method to the one described in this chapter.

There is, however, an improvement we can make. It is not necessary to calculate each $\mathcal{R}$-class from scratch as will become apparent if we consider the following lemma.

Lemma 5.4.1. *Suppose that at some point in the Orbit Algorithm $(u, T_u) \in R$ and there exists $x \in X \cup X^{-1}$. Then $xu\mathcal{L}u$ if and only if **Path**$((u, T_u), x^{-1}) = True$.*

PROOF: Suppose $xu\mathcal{L}u$. We know that

$$u^{-1}u = (u^{-1}x^{-1})(xu),$$

pre-multiplying by u we get

$$u = (uu^{-1})(x^{-1}x)u = (x^{-1}x)(uu^{-1})u = x^{-1}xu,$$

therefore **Path**$((u, T_u), x^{-1}xu) = True$ and in particular **Path**$((u, T_u), x^{-1}) = True$.

Let w label a (uu^{-1}, u)-walk in Γ_u. Suppose **Path**$((u, T_u), x^{-1}) = True$, then clearly $T_u(1, x^{-1}x) = 1$ and so $T_u(1, x^{-1}xw) = T_u(1, w)$ and so $u = x^{-1}xu$ and $xu\mathcal{L}u$ as required. ■

If $xu\mathcal{L}u$ as above then by Theorem 4.2.11 we know that there is a word graph isomorphism from $\phi : \Gamma_u \to \Gamma_{xu}$ with the condition that $u\phi = xu$ and hence ϕ will map the start (ie. uu^{-1}) of (uu^{-1}, Γ_u, u) to xuu^{-1}. Converting this to the tables we see that all we have done is move the starting vertex of T_u to the x^{-1} entry to create T_{xu}. Hence, when the Orbit Algorithm has found an $\mathcal{R}$-class, R_u, in a certain $\mathcal{D}$-class, D_u, then it never has to enumerate any of the other $\mathcal{R}$-classes in D_u. This does not mean however that the other $\mathcal{R}$-classes in D_u do not have to be checked against R_u as shifting the start does not necessarily create a new $\mathcal{R}$-class.

The final version written in GAP has this improvement (see Appendix).

Just as in groups we are interested in permutation representatives for generators and in monoids we are interested in transformation representatives for generators, for inverse monoids we are interested in partial injection representatives for generators. It is apparent that the Orbit Algorithm produces a collection of tables rather than a single table where information can be read off. It is now clearly no problem to find the number of elements in a finite inverse monoid as this number is merely the sum of the numbers of elements of $\mathcal{R}$-classes. Fortunately to find a partial injection representation for a generator is similarly no great problem, all we need to do is append the tables together (while renumbering the cosets so as to distinguish between the cosets of one $\mathcal{R}$-class from another) to get one big table, the *fully appended table*, and then read off from the columns to get partial injection for each generator and its inverse (entries in the table which read $T(c, x) = \perp$ simply mean that c is not included in the generator x's domain). To show this, recall the notation and ideas of **Section 2.4** on the Wagner representation theorem and consider the following theorem:

Theorem 5.4.2. *Let S be an inverse semigroup. For each $a \in S$ and for each $\mathcal{R}$-class R_u in S, we construct a partial injection as follows:*

$$p^a|_{R_u} : x \mapsto xa, (x \in R_u \cap R_u a^{-1})$$

$$p^a|_{R_u} : x \mapsto 0, (x \in R_u \backslash R_u a^{-1}).$$

For each $a \in S$, we define the partial symmetry on S as follows:

$$q^a : x \mapsto xp^a|_{R_x}.$$

The mapping

$$q : a \mapsto q^a, (a \in S)$$

is a monomorphism of S into $\mathcal{I}(S)$.

PROOF: The difference between this theorem and the Wagner representation theorem (Theorem 2.4.5) is that for each $a \in S$ in the latter we have $xw^a = xa$ when $x \in Sa^{-1}$ and $xw^a = 0$ otherwise while in the former $xq^a = xp^a|_{R_x} = xa$ when $x \in R_x a^{-1}$ and $xq^a = xp^a|_{R_x} = 0$ when $x \notin R_x a^{-1}$ (obviously x is always a member of R_x).

These constructions are identical because if $x \in Sa^{-1} = Saa^{-1}$, then $x = yaa^{-1}$ for some $y \in S$. Clearly x is a right multiple of ya and likewise $xa = (ya)a^{-1}a = ya$ and so ya is a right multiple of x and so $ya \in R_x$, hence $x = yaa^{-1} \in R_x a^{-1}$ as required. Clearly if $x \notin Sa^{-1}$ then in particular $x \notin R_x a^{-1}$. ■

This theorem basically says that the generators will map cosets only to cosets in the same $\mathcal{R}$-class, that is $\mathcal{R}$-classes form *blocks*. Moreover this behaviour is implicit in the Wagner Representation Theorem.

EXAMPLE: Let M be the inverse monoid presented by $\langle x | x^4 = x^2 \rangle$ then the fully appended table would look like:

Cosets	x	x^{-1}	Block
1	$\perp$	$\perp$	R_ϵ
2	3	$\perp$	R_x
3	$\perp$	2	R_x
4	5	$\perp$	$R_{x^{-1}}$
5	$\perp$	4	$R_{x^{-1}}$
6	7	7	R_{x^2}
7	6	6	R_{x^2}

5.5 Right Congruences

As we saw in **Chapter 1**, in general group Todd-Coxeter enumerates the cosets of a subgroup rather than simply the entire group. We saw how to use a right congruence to do this. This idea extends partially but readily to inverse monoid enumeration. In group Todd-Coxeter we found transversals for subgroups as right congruence classes, this is precisely what we can do in the inverse monoid enumerator.

For the rest of this section M is an inverse monoid, $u \in M$ and $\zeta \subseteq |^M_{R_u}$ is a right congruence on M.

Given that ζ is generated by $A = \{(r_i, s_i) | r_i, s_i \in (X \cup X^{-1})^*, i \in I\}$ for some index set I we define $A_u = \{(r_i, s_i) | r_i \geq u\}$. Note that by Corollary 4.2.7 (ii) for each $(r, s) \in A_u$, both r and s label a path in Γ_u.

Lemma 5.5.1. *Given $v \in M$ then $v \in R_u$ if and only if $v\zeta \in R_u/\zeta$.*

PROOF: The direct implication is immediate. Suppose that $v\zeta \in R_u/\zeta$ then $v\zeta\mathcal{R}u\zeta$ and as $\zeta \subseteq |_{R_u}^M$ then $v\mathcal{R}u$ and hence $v \in R_u$. ∎

Corollary 5.5.2. *There is a word graph epimorphism $\phi : \Gamma_u \to \Gamma_u/\zeta$ and so $L[(uu^{-1}, \Gamma_u, u)] \subseteq L[(uu^{-1}\zeta, \Gamma_u/\zeta, u\zeta)]$.*

PROOF: For $v \in V(\Gamma_u)$ define $v\phi = v\zeta$. For $(v_1, x, v_2) \in E(\Gamma_u)$ define $(v_1, x, v_2)\phi = (v_1\zeta, x, v_2\zeta)$. This mapping is clearly well-defined and preserves labeling. By Lemma 5.5.1 we can see that it is also onto.

$L[(uu^{-1}, \Gamma_u, u)] \subseteq L[(uu^{-1}\zeta, \Gamma_u/\zeta, u\zeta)]$ follows from Lemma 4.1.8. ∎

From Lemma 5.5.1 we can conclude that $(r, s) \in A_u$ if and only if $r\zeta\mathcal{R}u\zeta$. Hence if ζ is the right congruence generated by A_u then $R_u/\zeta = R_r/\zeta$.

The $\mathcal{R}$-class Algorithm can now be modified to enumerate these right congruence classes. At the beginning of the Repeat loop in the **Main Procedure** we insert the following:

- For each $(r, s) \in A_u$ do the following
  - **Trace**$(1, (r, s))$

This is called the *Generalised $\mathcal{R}$-class Algorithm.*

Right congruences are treated in exactly the same way as relations except that the starting point for any elementary $\mathcal{P}$-expansion is the idempotent uu^{-1}, this is because if $(r, s) \in \zeta$ then both $uu^{-1}r$ and $uu^{-1}s$ are $\mathcal{R}$ related to uu^{-1} and as $\zeta \subseteq \mathcal{R}$, $(uu^{-1}r)\zeta = (uu^{-1}s)\zeta$, while for any other $v \in R_u$ we do not know *a priori* if $(vr)\zeta = (vs)\zeta$.

The proof that the modified $\mathcal{R}$-class Algorithm terminates if and only if R_u/ζ is finite is exactly the same as the proof that the $\mathcal{R}$-class Algorithm terminates if and only if R_u is finite.

In a certain sense these results are not terribly exciting. The cases where $\zeta \not\subseteq |_{R_u}^M$ are surely much richer, but far more difficult to deal with - if it is possible to find a general solution at all. The power of the Inverse Monoid Enumerator comes from the fact that it separates right multiplication within $\mathcal{R}$-classes away from left multiplication of those $\mathcal{R}$-classes as blocks and so structures which "cut across" $\mathcal{R}$-classes fundamentally interfere with this process. Not only that, but it remains problematic deciding whether or not a right congruence is a subset of $\mathcal{R}$. Having said that though, being able to factor out large or even infinite substructures is incredibly useful just as it is in group theory. In particular it is useful to be able to factor out H_u which I shall look at in the next section.

5.6 Enumerating R_u/H_u

We have the following lemma.

Lemma 5.6.1. *Given an idempotent $e \in M$ and $v \in R_e$ then $v \in H_e$ if and only if $ev = ve$ and $e = ev^{-1}v$.*

PROOF: Suppose $v \in H_e$. By Corollary 2.1.7, H_e is a group with e as an identity and so by the group axioms $ev = ve$ and $e = ev^{-1}v$.

Conversely suppose that $ev = ve$ and $e = ev^{-1}v$, then we want to show that $e\mathcal{L}v$. Now as $v = ev$ then $ve = ev = v$ and $v^{-1}v = v^{-1}ev = v^{-1}ve = ev^{-1}v = e$ as required. ■

Given $u \in M$ it is now possible to modify the Generalised $\mathcal{R}$-class Algorithm so that it constructs the smallest right congruence ζ_H on M such that

$$H_{uu^{-1}} \times H_{uu^{-1}} \subseteq \zeta_{H_u} \subseteq |^M_{R_u}.$$

This is done by giving each $c \in C$ a normal form $N(c) \in (X \cup X^{-1})^*$ and inserting **Trace** from **Section 5.1** and the following subroutine into the Generalised $\mathcal{R}$-class Algorithm.

Find H

- Parameters: None
- Local: c

For $c \in C$ do

- If $r(c) = 0$ And **Trace**$(N(c)u) \neq \perp$ And **Trace**$(N(c)^{-1}N(c)) \neq \perp$ Then
  - Add $(N(c), \epsilon)$ to A_u

$N(c)$ is calculated very easily in the **New** subroutine. Firstly $N(1) := \epsilon$ and when coset $d \in C$ is defined as the entry $T(c, x)$ then $N(d) := N(c)x$.

This R_u/H_u Algorithm will now construct generators for the right congruence for ζ_{H_u}. Essentially what happens is that the group $H_{uu^{-1}}$ is factored out of R_u and each coset represents an $\mathcal{H}$-class in R_u. Note that given a right congruence ζ there is nothing to stop this algorithm enumerating R/ζ' where ζ' is the right congruence generated by $\zeta_{H_u} \cup \zeta$. This done by just adding the generators for ζ into A_u at the start of the algorithm.

The result of this algorithm will give the action of R_u on $H_{uu^{-1}}$: that is the structure of the $\mathcal{H}$-classes in R_u. The word graph for this is denoted $S\Gamma(u)/H_u$. It should be noted that if $S\Gamma(u)/H_u$ is isomorphic to $S\Gamma(v)/H_v$ then it does not necessarily follow that $u\mathcal{R}v$.

By N. Ruškuc [23] we know that ζ_{H_u} is finitely generated, moreover this algorithm systematically creates pairs for ζ_{H_u} and so as there is a finite generating set for ζ_{H_u}, the algorithm will terminate.

Chapter 6

More on Inverse Monoids

In this chapter I explain my work with Allessandra Cherubini and Brunnetto Piochi, looking at some of the applications of the algorithm described in **Chapter 5**. Not all theorems are original, the results in **Section 6.1** for example can be found in Petrich [18]. However the results are all proved in an original manner using the insights from **Chapter 4** and **Chapter 5** to directly tackle inverse monoid presetation theory questions.

Most of this chapter is about presentations and the following concepts will be needed.

Firstly, though, I shall introduce (or restate) some notation that I shall use throughout this chapter.

NOTATION: The greatest common divisor of the positive integers $r_1, r_2, ...$ is denoted $(r_1, r_2, ...)$.

NOTATION: The cyclic permutation of the objects $x_1, x_2, ..., x_n$ is denoted $(x_1 x_2 ... x_n)$.

NOTATION: Given a word $w \in (X \cup X^{-1})^*$ then $\overline{w}$ is the free cancellation of w in $\mathbf{F}_G(X)$.

I shall introduce the following theorem which is very useful for discussing inverse monoid and inverse semigroup presentations in general.

Theorem 6.0.2. *Let M be an inverse monoid presented by $\langle X | U \rangle$ with an idempotent e such that $ef = e$ for every idempotent $f \in M$. Then R_e is isomorphic to the group $(X \cup X^{-1})^* / \sigma$ where σ is the group congruence generated by U.*

PROOF: Given any $w \in M$ then $e\mathcal{R}ew$ as $e = eww^{-1}$ and similarly $e\mathcal{L}ew$ as $e = ew^{-1}w = w^{-1}we$ and so $ew \in H_e$. Therefore $eM = Me = H_e$ which is a group. Clearly H_e is a homomorphic image of M as for each $(u, v) \in U$ then $eu = ev$.

Given any $(s,t) \in \sigma$ then $\overline{s}\tau = \overline{t}\tau$ where τ is the inverse monoid congruence generated by U therefore

$$e(s\tau) = e(\overline{s}\tau) = e(\overline{t}\tau) = e(t\tau)$$

and conversely by Corollary 4.2.8 (iii) if $e(s\tau) = e(t\tau)$ then $s\sigma = t\sigma$. All that remains to do is to construct the isomorphism $\phi : (X \cup X^{-1})^*/\sigma \to eM$ by defining $\phi : s\sigma \mapsto e(s\tau)$. ∎

6.1 Monogenic Inverse Monoids

Let us start by looking at the most simple example of inverse monoids.

Definition 6.1.1. Let $\mathcal{V}$ be a variety and let $O \in \mathcal{V}$. We say that O is *monogenic* if it is generated by a single element $x \in O$. In other words there exists a unique homomorphism from $\mathrm{F}_{\mathcal{V}}(X)$ to M where X contains one symbol. In particular we have *monogenic monoids*, *monogenic inverse monoids* and *monogenic groups*.

Monogenic monoids are easily to characterised by the following well-known theorem.

Theorem 6.1.2. *Let M be a monogenic monoid generated by x then M is presented by $\langle x | x^m = x^n\rangle$ for some distinct non-negative integers m and n. Furthermore $|M| = \max(m,n)$ if $m \neq n$ and $M = \{x\}^*$ if $m = n$.*

In other words monogenic monoids are either free or *one-relation monoids*. Similarly monogenic semigroups and monogenic groups are either free or *one-relation semigroups* and *one-relation groups* respectively. Monogenic groups are usually called *cyclic groups*.

EXAMPLE: Given the monoid M presented by $\langle x | x^5 = x^2\rangle$ then the Cayley graph is:

$$\begin{tikzcd}
 & & & x^4 \arrow[dl] & \\
\epsilon \arrow[r] & x \arrow[r] & x^2 \arrow[rr] & & x^3 \arrow[ul]
\end{tikzcd}$$

Essentially monogenic monoids are "tadpoles" in that they are cyclic groups with a "tail". Likewise the semigroup given by the same presentation has the following Cayley graph tadpole:

$$\begin{tikzcd} & & x^4 \arrow[dl] & \\ x \arrow[r] & x^2 \arrow[rr] & & x^3 \arrow[ul] \end{tikzcd}$$

Whereas the group given by the same presentation has a Cayley graph which is the determinised form for the Cayley graph of the semigroup (or the monoid).

$$\begin{tikzcd} & x^2 \arrow[dl] & \\ \epsilon \arrow[rr] & & x \arrow[ul] \end{tikzcd}$$

It is generally true that Cayley graphs for groups, like Schützenberger graphs for $\mathcal{R}$-classes of inverse monoids, are deterministic and injective whereas Cayley graphs for monoids are only deterministic.

We now turn our attention to monogenic inverse monoids. From here on, ρ shall be the free inverse monoid congruence on $\{x, x^{-1}\}^*$ and M shall be the monogenic inverse monoid $\{x, x^{-1}\}^*/\tau$ for some congruence $\tau \supseteq \rho$ on $\{x, x^{-1}\}^*$.

The case of monogenic inverse monoids is somewhat more complex. If we consider a monogenic inverse monoid $M = \langle x|U\rangle$ then M is presented by $\langle x, x^{-1}|U \cup \rho\rangle$ as a monoid. M is therefore not necessarily a monogenic monoid, unless, that is, it is possible to eliminate one of the two generators - for example if $x^{-1} = x^2$ is a consequence of U. Let us distinguish between two types of inverse monoid relations.

Definition 6.1.3. Given an inverse monoid presentation $P = \langle X|U\rangle$. The relation $(u, v) \in U$ is an *idempotent relation* if $\overline{u} = \overline{v}$. Conversely we have *non-idempotent relations* where $\overline{u} \neq \overline{v}$. The relation $(u, v) \in U$ is *trivial* if $u \equiv v$. Similarly if $(X \cup X^{-1})^*/\tau$ is an inverse monoid then an equation $u\tau = v\tau$ $(u, v \in (X \cup X^{-1})^*)$ is called an *idempotent equation* if $\overline{u} = \overline{v}$. The equation $u\tau = v\tau$ is *trivial* if $u \equiv v$.

Idempotent relations are quite special to inverse monoid and inverse semigroup presentations. They are always trivial in monoid presentations and can always be freely reduced to trivial relations in group presentations. Note that given an idempotent relation (u, v) in a presentation $\langle X|U\rangle$ for M, it is possible that both

$u\tau$ and $v\tau$ are non-idempotent in M, however if one of u or v is idempotent then the other one is as well.

At this point we shall recall that it was shown in **Chapter 3** that we can write $u \in \mathbf{F}_{\mathcal{IM}}(x)$ as $(x^{-m}x^{m+n}x^{-n}x^{k})\rho$ where $m, n \geq 0$, $-m \leq k \leq n$ and we know that u is idempotent if and only if $k = 0$. Alternatively we can write u as its free group representative which will be

$$FG(u\rho^{-1}) = (\{x^{-m}, x^{-m+1}, ..., \epsilon, x, ..., x^{n}\}, x^{k})$$

It is easy to see that $FG(u\rho^{-1}) = FG(x^{-m}x^{m+n}x^{n-k})$ and we may use the representative $x^{-m}x^{p}x^{-q}$ where $p = m + n$ and $q = k - n$ and conversely $n = p - m$ and $k = -m + p - q$. The condition on m, p and q is $0 \leq m, q \leq p$.

Now, suppose that we have $u_1, u_2 \in \mathbf{F}_{\mathcal{IM}}(x)$ with free group representations $(\{x^{-m_1}, ..., x^{n_1}\}, x^{k_1})$ and $(\{x^{-m_2}, ..., x^{n_2}\}, x^{k_2})$ then

$$\begin{aligned} FG(u_1u_2) &= (\{x^{-m}, ..., x^{n}\} \cup x^{k}\{x^{-m_2}, ..., x^{n_2}\}, \overline{x^{k_1}x^{k_2}}) \\ &= (\{x^{\min(-m_1, k_1-m_2)}, ..., x^{\max(n_1, k_1+n_2)}\}, x^{k_1+k_2}) \end{aligned}$$

In other words

$$\begin{aligned} &(x^{-m_1}x^{p_1}x^{q_1})(x^{-m_2}x^{p_2}x^{q_2}) \\ &= x^{\min(-m_1, k_2-m_2)}x^{\max(p_1-m_1, -m_1+p_1-q_1+p_2-m_2)}x^{q_1+p+1-m_1+q_2+p_2-m_2} \end{aligned}$$

Lemma 6.1.4. *If the equation*

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

with $0 \leq m_1, q_1 \leq p_1$, $0 \leq m_2, q_1 \leq p_2$ *and* $p_1 \geq p_2$ *holds in* M *then*

$$x^{p_2+|k_1-k_2|}\tau = x^{p_2}\tau$$

where $k_1 = -m_1 + p_1 - q_1$ *and* $k_2 = -m_2 + p_2 - q_2$.

PROOF: We know that in M,

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau. \tag{6.1}$$

Multiplying 6.1 on the right by $x^{q_2}\tau$ and on the left by $x^{m_2}\tau$ and noticing that both $q_2 \leq p_2$ and $m_2 \leq p_2$ so we can cancel the right hand side of the equation, then we get

$$(x^{m_2}x^{-m_1}x^{p_1}x^{-q_1}x^{q_2})\tau = x^{p_2}\tau. \tag{6.2}$$

Now the free group representative of the left hand side of 6.2, $u_l \in \mathbf{F}_{\mathcal{IM}}(x)$, is

$$FG(u_l) = (\{x^{\min(0,m_2-m_1)}, ..., x^{m_2-m_1+p_1+\max(0,q_2-q_1)}\}, x^{k_1+m_2+q_2})$$

while the free group representative of the right hand side of 6.2, $u_r \in \mathbf{F}_{\mathcal{IM}}(x)$ is

$$FG(u_r) = (\{\epsilon, x, ..., x^{p_2}\}, x^{p_2}).$$

Now clearly $u_l\tau \in R_{u_r\tau}$ and so

$$u_l\tau = (u_r {u_r}^{-1}\overline{u_l})\tau = (x^{p_2}x^{-p_2}x^{k_1+m_2+p_2})\tau = (x^{p_2}x^{-p_2}x^{k_1-k_2+p_2})\tau.$$

If $k_1 \geq k_2$ then

$$u_l\tau = x^{p_2+k_1-k_2}\tau$$

as required, otherwise

$$\begin{aligned} {u_l}^{-1}\tau &= {u_r}^{-1}\tau = x^{-p_2}\tau = (x^{-p_2}x^{p_2}x^{-p_2})\tau \\ &= (x^{-p_2}u_l x^{-p_2})\tau = (x^{-p_2}x^{p_2}x^{-p_2}x^{p_2+k_1-k_2}x^{-p_2})\tau \\ &= x^{-p_2+k_1-k_2}\tau \end{aligned}$$

and so

$$u_l\tau = x^{p_2+k_1-k_2}\tau$$

as required. ■

We want to show that

$$\langle x | x^{-m_1}x^{p_1}x^{-q_1} = x^{-m_2}x^{p_2}x^{-q_2}\rangle$$

is equivalent to

$$\langle x | x^{p_2+|k_1-k_2|} = x^{p_2}\rangle,$$

however it is only true when the relation is non-idempotent. We need the following lemma.

Lemma 6.1.5. *If the equation*

$$x^{p+k}\tau = x^p\tau$$

with $p \geq 0$ and $k > 0$ holds in M then $R_{x^p\tau}$ is the group $\{x, x^{-1}\}^*/\sigma$ where σ is the minimum group congruence. $R_{x^p\tau}$ is a group homomorphic image of the cyclic group C_k.

PROOF: Clearly x^p and x^{p+k} label $(x^p x^{-p}\tau, x^p\tau)$-walks in $S\Gamma(x^p\tau)$ and so by Corollary 4.2.8 (iii), $x^p\sigma = x^{p+k}\sigma$ where σ is the minimum group congruence. Therefore $(x^k, \epsilon) \in \sigma$ and so $\{x, x^{-1}\}^*/\sigma$ is a homomorphic image of C_k as required. ■

Lemma 6.1.6. *Let $M = \{x, x^{-1}\}^*/\tau$ be a monogenic inverse monoid generated by x. If the equation*

$$x^{p+k}\tau = x^p\tau$$

with $p \geq 0$ and $k > 0$ holds in M then

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

where $p_1 \geq p_2 = p$ and $|-m_1+p_1-q_1+m_2-p_2+q_2| = k$ and $0 \leq m_1, q_1 \leq p_1$ and $0 \leq m_2, q_2 \leq p_2$.

PROOF: We know that in M,

$$x^{p+k}\tau = x^p\tau. \tag{6.3}$$

By Lemma 6.1.5, $R_{x^p\tau}$ is a homomorphic image of the cyclic group C_k with $e = (x^p x^{-p})\tau$ as the identity and generated by $x^p x^{-p} x\tau$.

Multiplying 6.3 on the left by $x^{-m_2}\tau$ and on the right by $x^{-q_2}\tau$ (where $0 \leq m_2, q_2 \leq p_2$) and substituting p_2 for p, we have

$$(x^{-m_2}x^{p+k}x^{-q_2})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau.$$

We simply note that it is possible to cancel on both sides of this equation on both the left and the right as the elements are products of members of the cyclic group $\{x, x^{-1}\}^*/\sigma$ and that $x^p = x^{p+lk}$ for any $l \geq 0$. In this case pick a $p_1 \geq p_2$ and $0 \leq m_1, q_1 \leq p_1$ so that $-m_1 + p_1 - q_1 = p + k$ and we may construct the result. ■

The previous lemma breaks down when $k = 0$ and we have the trivial equation $x^p\tau = x^p\tau$. We thus turn our attention to idempotent equations. We already know about the infinite monogenic inverse monoid $\mathbf{F}_{\mathcal{IM}}(x)$, here I introduce another example.

Definition 6.1.7. The *bicyclic monoid generated by x, B_x,* is the monoid presented by $\langle x | xx^{-1} = \epsilon \rangle$.

There is a dual of B_x, denoted by ${B_x}^1$ which is presented by $\langle x | x^{-1}x = \epsilon \rangle$.

Throughout the rest of this section I shall denote the congruence generated by $\rho \cup (xx^{-1}, \epsilon)$ by υ and I shall denote the congruence generated by $\rho \cup (x^{-1}x, \epsilon)$ by υ^1.

Lemma 6.1.8. *If $w \in \{x, x^{-1}\}^*$ so that $w\rho = (x^{-m}x^p x^{-q})\rho$ $(0 \leq m, q \leq p)$ then $w\upsilon = (x^{-m}x^{p-q})\upsilon$ and dually $w\upsilon^1 = (x^{p-m}x^{-q})\upsilon^1$.*

PROOF: It is clear from the definition of υ that we may cancel w on the right. Similarly from the definition of υ^1 we may cancel w on the left. We need only note that $p \geq q$ and $p \geq m$ to get each result. ■

Lemma 6.1.9. *Suppose that $(u_1, u_2) \in \upsilon$ with $FG(u_1) = (\{x^{-m_1}, ..., x^{n_1}\}, x^{k_1})$ and $FG(u_2) = (\{x^{-m_2}, ..., x^{n_2}\}, x^{k_2})$ then $m_1 = m_2$ and $k_1 = k_2$. Dually suppose that $(u_1, u_2) \in \upsilon^1$ then $n_1 = n_2$ and $k_1 = k_2$.*

PROOF: We already know that $FG(u) = FG(v)$ for all $(u, v) \in \rho$. Now $FG(xx^{-1}) = (\{\epsilon, x\}, \epsilon)$ while $FG(\epsilon) = (\{\epsilon\}, \epsilon)$ and $(xx^{-1})^{-1} = xx^{-1}$ and $\epsilon^{-1} = \epsilon$ and so the result holds for all the generators and their inverses of υ. Now

$$FG(u_1u_2) = (\{x^{\min(-m_1, k_1 - m_2)}, ..., x^{\max(n_1, k_1 + n_2)}, x^{k_1 + k_2})$$

and if we notice that $\min(-m_1, k_1 - m_2)$ and $k_1 + k_2$ are not dependent on either n_1 or n_2, we have our result.

The dual is proved in the same way. ■

Lemma 6.1.10. *In B_x, the Schützenberger graph $S\Gamma(\epsilon\upsilon)$ has vertices*

$$V(S\Gamma(\epsilon\upsilon)) = \{x^s\upsilon | s \geq 0\}$$

and edges

$$E(S\Gamma(\epsilon\upsilon)) = \{(x^s\upsilon, x, x^{s+1}\upsilon), (x^{s+1}\upsilon, x^{-1}, x^s\upsilon) | s \geq 0\}$$

The set of idempotents in B_x is $\{(x^{-r}x^r)\upsilon | r \geq 0\}$ and each $R_{(x^{-r}x^r)\upsilon}$ have Schützenberger graphs isomorphic to $S\Gamma(\epsilon\upsilon)$.

PROOF: $\epsilon\upsilon x^s\upsilon = x^s\upsilon$ and $x^s\upsilon x^{-s}\upsilon = \epsilon\upsilon$ and so $x^s\upsilon\mathcal{R}\epsilon\upsilon$.

By Lemma 6.1.9 $x^s\upsilon = x^t\upsilon$ if and only if $s = t$ and noticing that $x^s\upsilon x\upsilon = x^{s+1}\upsilon$ and $x^{s+1}\upsilon x^{-1}\upsilon = x^s\upsilon$ ($s \geq 0$) we can conclude that $S\Gamma(\epsilon\upsilon)$ is as described.

The set of idempotents of $\{x, x^{-1}\}^*/\rho$ is $\{(x^{-r_1}x^{r_1+r_2}x^{-r_2})\rho | r_1, r_2 \geq 0\}$, and by Lemma 6.1.8 $(x^{-r_1}x^{r_1+r_2}x^{-r_2})\upsilon = (x^{-r_1}x^{r_2})\upsilon$. Moreover $(x^{-r_1}x^{r_1})\upsilon \neq (x^{-r_2}x^{r_2})\upsilon$ if $r_1 \neq r_2$ and so $r_1 = r_2$ if $x^{-r_1}x^{r_1}\upsilon = x^{-r_2}x^{r_2}\upsilon$ by Lemma 6.1.9. Hence all the idempotents are distinct.

Let $\phi : R_{\epsilon\upsilon} \to R_{(x^{-r}x^r)\upsilon}$ by $\phi : x^s \mapsto x^{-r}x^s$. Let $\phi' : R_{(x^{-r}x^r)\upsilon} \to R_{\epsilon\upsilon}$ by $\phi' : x^{-r}x^s \mapsto x^s$. It is clear that ϕ and ϕ' are mutually inverse and induce word graph homomorphisms on the respective Schützenberger graphs. ■

For the dual of the above, in ${B_x}^1$, the Schützenberger graph $S\Gamma(\epsilon\upsilon^1)$ is simply $S\Gamma^1(\epsilon\upsilon)$ in B_x.

In B_x, $S\Gamma(x^{-r}x^r\upsilon)$ looks like this:

$$\begin{array}{ccccccccc} x^{-r} & \to_x & x^{-r}x & \to_x \cdots & \to_x & x^{-r}x^r & \to_x & x^{-r}x^{r+1} & \to_x \cdots \\ & & & & & \updownarrow & & & \end{array}$$

NOTATION: We denote the $\mathcal{R}$-class of B_x which contains ϵ by R_{B_x} and the Schützenberger graph of R_{B_x} by $S\Gamma_{B_x}$.

Lemma 6.1.11. *If the equation*

$$(x^{p+q}x^{-q})\tau = x^p\tau$$

with $p \geq 0$ and $q > 0$ holds in M then $S\Gamma((x^{-r}x^s)\tau)$ with $r, s \geq 0$ is a homomorphic image of $S\Gamma_{B_x}$ if either $r > p$ or $s > p$.

Dually if the equation

$$(x^{-q}x^{p+q})\tau = x^p\tau$$

with $p \geq 0$ and $q > 0$ holds in M then $S\Gamma((x^rx^{-s})\tau)$ with $r, s \geq 0$ is a homomorphic image of $S\Gamma^1{}_{B_x}$ if either $r > p$ or $s > p$.

PROOF: The idempotent in $R_{(x^{-r}x^s)\tau}$ is $(x^{-r}x^sx^{-s}x^r)\tau$. Suppose $r > p$ then for $k \geq 0$

$$\begin{aligned} (x^{-r}x^sx^{-s}x^rx^k)\tau(x^{-k}\tau) &= (x^{-r}x^sx^{-s}(x^{r+kq}x^{-kq})x^kx^{-k})\tau \\ &= (x^{-r}x^sx^{-s}x^{r+kq}x^{-kq})\tau \\ &= (x^{-r}x^sx^{-s}x^r)\tau \end{aligned}$$

and so $(x^{-r}x^sx^{-s}x^rx^k)\tau\mathcal{R}(x^{-r}x^sx^{-s}x^r)\tau$. Likewise if $s > p$, then for $k \geq 0$

$$\begin{aligned}(x^{-r}x^sx^{-s}x^rx^k)\tau x^{-k}\tau &= (x^{-r}(x^{s+(k+r)q}x^{-s-(k+r)q})x^rx^kx^{-k})\tau \\ &= (x^{-r}x^{s+(k+r)q}x^{-s-(k+r)q}x^{r+k}x^{-k})\tau \\ &= (x^{-r}x^{s+(k+r)q}x^{-s-(k+r)q}x^r)\tau \\ &= (x^{-r}x^sx^{-s}x^r)\tau\end{aligned}$$

and so $(x^{-r}x^sx^{-s}x^rx^k)\tau\mathcal{R}(x^{-r}x^sx^{-s}x^r)\tau$.

Define the mapping $\phi : R_{B_x} \to R_{(x^{-r}x^s)\tau}$ by $\phi : x^k v \mapsto (x^{-r}x^sx^{-s}x^rx^k)\tau$ $(k \geq 0)$. It is easy to see that ϕ preserves labelling and orientation of edges.

The dual result is proved in the same way. ∎

Corollary 6.1.12. *If the equation*

$$(x^{p+q}x^{-q})\tau = x^p\tau$$

where $p \geq 0$ *and* $q > 0$ *holds in* M *then*

$$(x^{p+q'}x^{-q'})\tau = x^p\tau$$

holds in M *for every* $q' > 0$.

Dually if the equation

$$(x^{-q}x^{p+q})\tau = x^p\tau$$

where ≥ 0 *and* $q > 0$ *holds in* M *then*

$$(x^{-q'}x^{p+q'})\tau = x^p\tau$$

holds in M *for every* $q' \geq 0$.

PROOF: Suppose that

$$(x^{p+q}x^{-q})\tau = x^p\tau.$$

Consider $R_{x^p\tau}$. We can see that from the proof of Lemma 6.1.11 that there is an $((x^px^{-p})\tau, (x^px^{-p}x^k)\tau)$-walk in $S\Gamma(x^p\tau)$ for every $k \geq 0$. In particular if $k = p + q'$ $(q' \geq 0)$, then we have

$$(x^px^{-p}x^{p+q'})\tau\mathcal{R}(x^px^{-p})\tau$$

that is

$$(x^px^{-p}x^{p+q'}x^{-p-q'})\tau = (x^px^{-p})\tau$$

which cancels down to

$$(x^{p+q'}x^{-p-q'})\tau = (x^p x^{-p})\tau$$

multiply on the right by $x^p\tau$ and we have

$$(x^{p+q'}x^{q'})\tau = x^p\tau$$

as required.

The dual is proved in a similar manner. ∎

The other more obvious standard example of infinite monogenic inverse monoids is the infinite cyclic group $\mathbf{F}_G(x)$.

Lemma 6.1.13. *If the idempotent equation*

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

with $0 \le m_1, q_2 \le p_1$, $0 \le m_2, q_2 \le p_2$, $m_1 \ne m_2$, $q_1 \ne q_2$ *and* $p_1 \ge p_2 > 0$ *holds in* M *then* $R_{x^{p_2}\tau}$ *is a group.*

PROOF: Now $(x^{p_2}x^{-p_2})\tau$ is the idempotent in $R_{x^{p_2}\tau}$ we need to show that

$$((x^{p_2}x^{-p_2})(x^{-m}x^p x^{-q}))\tau = (x^{p_2}x^{-p_2}x^{-m+p-q})\tau$$

for any $0 \le m, q \le p$. Now

$$\begin{aligned} x^{p_2}\tau &= (x^{m_2}x^{-m_2}x^{p_2}x^{-q_2}x^{q_2})\tau \\ &= (x^{m_2}x^{-m_1}x^{p_1}x^{-q_1}x^{q_2})\tau. \end{aligned}$$

I shall split this equation up into three cases

1. $m_1 > m_2$, $q_1 > q_2$ In this case

   $$x^{p_2}\tau = (x^{m_2-m_1}x^{p_1}x^{q_2-q_1})\tau$$

   and so

   $$\begin{aligned} (x^{p_2}x^{-p_2})\tau &= (x^{m_2-m_1}x^{p_1}x^{m_1-m_2-p_1}\tau \\ &= (x^{m_2-m_1}x^{p_2}x^{-p_2}x^{p_1}x^{m_1-m_2-p_1}\tau \end{aligned}$$

   and if we substitute $(x^{p_2}x^{-p_2})\tau$ back into the above k times we get

   $$\begin{aligned} (x^{p_2}x^{-p_2})\tau &= (x^{k(m_2-m_1)}x^{p_2}x^{-p_2}x^{p_1+k(m_1-m_2)}x^{m_1-m_2-p_1}\tau \\ &= (x^{k(m_2-m_1)}x^{kp_1+(k-1)(m_1-m_2-p_1)}x^{m_1-m_2-p_1}\tau. \end{aligned}$$

   If we choose $k \ge 0$ such that $k(m_1 - m_2) \ge m$ and $p_1 + k(m_1 - m_2) \ge p$ and we have the result.

2. $m_1 > m_2,\ q_1 < q_2$ In this case

$$\begin{aligned} x^{p_2}\tau &= (x^{m_2-m_1}x^{p_1+q_2-q_1})\tau \\ &= (x^{m_2-m_1}x^{p_2}x^{p_1-p_2+q_2-q_1})\tau \end{aligned}$$

If we substitute x^{p_2} back in again k times we get

$$x^{p_2}\tau = (x^{k(m_2-m_1)}x^{p_2}x^{k(p_1-p_2+q_2-q_1)})\tau$$

and so all we need to do is choose $k \geq 0$ such that $k(m_2 - m_1) \geq m$ and $k(p_1 - p_2 + q_2 - q_1) \geq m$.

3. $m_1 > m_2,\ q_1 < q_2$ This is just the dual of case 2.

We need only note that the fourth case where $m_1 < m_2$ and $q_1 < q_2$ violates the condition that $p_1 \geq p_2$. ■

Clearly $R_{x^{p_2}\tau}$ is a homomorphic image of $\mathbf{F}_{\mathcal{G}}(x)$ ie. it is either free or cyclic.

Theorem 6.1.14. *The following statements are true.*

(i) *The idempotent equation*

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

where $0 \leq m_1, q_1 \leq p_1$, $0 \leq m_2, q_2 \leq p_2$, $m_1 = m_2$ *and* $p_1 - m_1 > p_2 - m_2$ *holds in* M *if and only if*

$$(x^{p_2+1}x^{-1})\tau = x^{p_2}\tau.$$

(ii) *The idempotent equation*

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

where $0 \leq m_1, q_1 \leq p_1$, $0 \leq m_2, q_2 \leq p_2$, $m_1 > m_2$ *and* $p_1 - m_1 = p_2 - m_2$ *holds in* M *if and only if*

$$(x^{-1}x^{p_2+1})\tau = x^{p_2}\tau.$$

(iii) *The idempotent equation*

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

where $0 \le m_1, q_1 \le p_1$, $0 \le m_2, q_2 \le p_2$, $m_1 \ne m_2$ *and* $p_1 - m_1 \ne p_2 - m_2$ *holds in* M *if and only if*

$$(x^{\min(p_1,p_2)}x^{-1})\tau = (x^{-1}x^{\min(p_1,p_2)})\tau.$$

PROOF:

(i) Suppose

$$(x^{-m_1}x^{p_1}x^{-q_1})\tau = (x^{-m_2}x^{p_2}x^{-q_2})\tau$$

with the conditions given. Now $m_1 = m_2$ and so multiplying on the left by m_1 and on the right by q_2 gives

$$(x^{p_1}x^{-q_1}x^{+q_2})\tau = x^{p_2}\tau.$$

By idempotency $q_1 = (p_1 - m_1) - (p_2 - m_2) + q_2 > q_2$ so

$$(x^{p_1}x^{-q})\tau = x^p\tau$$

where $p = p_2$ and $q = q_2 - q_1 > 0$. Hence by Corollary 6.1.12

$$(x^{p+1}x^{-1})\tau = x^p\tau.$$

Conversely suppose that

$$(x^{p+1}x^{-1})\tau = x^p\tau.$$

Let $m_1 = m_2 = q_2 = 0$, $p_1 = p + 1$, $p_2 = p$ and $q_1 = 1$ and the conditions are satisfied.

(ii) This is the dual case of (i).

(iii) This follows from Lemma 6.1.13

■

If we notice that $x^{p+k}\tau = x^p\tau \Rightarrow x^{p'+k}\tau = x^{p'}\tau$ if $p' \ge p$ and $x^{p+k}\tau = x^p\tau \Rightarrow (x^px^{-1})\tau = (x^{-1}x^p)\tau$ and that $(x^px^{-1})\tau = (x^{-1}x^p)\tau \Rightarrow (x^{p+1}x^{-1})\tau = x^p\tau$ we can now categorise monogenic inverse monoids. Either we have

1. M is presented by $\langle x | x^{p+k} = x^p \rangle$ where $p \geq 0$ and $k > 0$. In this case we have a chain of $\mathcal{D}$ classes D_{ϵ_T}, D_{x_T}, $D_{x^2{}_T}, ..., D_{x^p{}_T}$ where $D_{x^i{}_T}$ has $i+1$ $\mathcal{R}$-classes and $\mathcal{L}$-classes with each $\mathcal{H}$-class containing one element. For example $H_{x^{-m}x^{p'}x^{-q}} \subseteq R_{x^{-m}x^{p'}} \subseteq D_{x^{p'}}$ and dually $H_{x^{-m}x^{p'}x^{-q}} \subseteq L_{x^{p'}x^{-q}} \subseteq D_{x^{p'}}$ (for $p' < p$). The chain ends with D_{x^p} which is isomorphic to the cyclic group of order k. The order of M is

$$|M| = (\Sigma^p{}_{i=1} i^2) + k.$$

2. M is presented by $\langle x | x^{-1}x^p = x^p x^{-1} \rangle$ where $p > 0$. In this case we have the same chain of $\mathcal{D}$-classes as before except that D_{x^p} is isomorphic to $\mathbf{F}_{\mathcal{G}}(x)$.

3. M is presented by $\langle x | x^{p+1}x^{-1} = x^p \rangle$ where $p \geq 0$. In this case we have the same chain of $\mathcal{D}$-classes as before except that D_{x^p} is isomorphic to the bicyclic inverse monoid.

4. M is presented by $\langle x | x^{-1}x^{p+1} = x^p \rangle$ where $p \geq 0$. In this case we have the same chain of $\mathcal{D}$-classes as before except that D_{x^p} is isomorphic to the dual bicyclic inverse monoid.

5. Finally M can be free in which case the sequence of $\mathcal{D}$-classes continues indefinitely.

6.2 Coxeter Presentations

Definition 6.2.1. Let Υ be a finite directed graph with vertices with $V(\Upsilon) = X = \{x_1, x_2, ..., x_n\}$ where the vertex x_i is labelled by the positive integer p_i and the edge (x_i, x_j) is labelled by the positive integer q_{ij}. We call Υ a *Coxeter graph.* The *Coxeter presentation corresponding with* Υ is the semigroup presentation

$$\begin{aligned}\langle X| \quad & x_i{}^{p_i+1} = x_i \forall x_i \in X,\ (x_j x_k)^{q_{jk}} = x_j{}^{p_j} \forall (x_j, x_k) \in E(\Upsilon), \\ & x_j x_k = x_k x_j \forall (x_j, x_k) \notin E(\Upsilon)\rangle.\end{aligned}$$

Coxeter presentations can be considered as monoid, group, inverse monoid or inverse semigroup presentations. For the case of groups the presentation is equivalent to

$$\begin{aligned}\langle X| \quad & x_i{}^{p_i} = \epsilon\, \forall x_i \in X,\ (x_j x_k)^{q_{jk}} = \epsilon\, \forall (x_j, x_k) \in E(\Upsilon), \\ & x_j x_k = x_k x_j\ \forall (x_j, x_k) \notin E(\Upsilon)\rangle.\end{aligned}$$

Groups presented by Coxeter presentations are called *Coxeter groups* and similarly we have *Coxeter semigroups* and *Coxeter inverse semigroups.* As the presentations do not involve the identity, the monoid and inverse monoid cases are not particularly interesting and amount to merely adding an artificial identity to the semigroup. We shall look at some well known examples of Coxeter groups and semigroups.

EXAMPLE: The symmetric group S_n is a Coxeter group generated by $\{x_i = (i\, i+1)|1 \leq i \leq n-1\}$ with the group presentation

$$\langle X|{x_i}^2 = \epsilon(1 \leq i \leq n-1), (x_i x_{i+1})^3 = \epsilon, x_i x_j = x_j x_i (1 \leq i \leq n-2, |i-j| > 1)\rangle.$$

EXAMPLE: The dihedral group on $\{1, 2, ..., n\}$ is a Coxeter group generated by $x = (1\ 2...n)$ and $y = (1\ 2)$ with the group presentation

$$\langle x, y|x^n = y^2 = (xy)^2 = \epsilon\rangle$$

EXAMPLE: A finite group direct product of finite cyclic groups is a Coxeter group. Let G be the product

$$C_{p_1} \times C_{p_2} \times ... \times C_{p_n}$$

then G can be presented as a group by

$$\langle x_1, x_2, ..., x_n|{x_i}^{p_i} = \epsilon,\ x_j x_k = x_k x_j\ (1 \leq i, j, k \leq n)\rangle.$$

Note that the semigroup S given by the Coxeter presentation

$$\langle x_1, x_2, ..., x_n|{x_i}^{p_i+1} = x_i,\ x_j x_k = x_k x_j\ (1 \leq i, j, k \leq n)\rangle$$

is only a group when $n = 1$, in which case it is C_{p_1} with the identity being ${x_1}^{p_1}$.

Now we know from **Section 6.1** that the inverse semigroup given by the Coxeter inverse semigroup presentation

$$\langle x|x^{p+1} = x\rangle$$

is just the cyclic group of order p with the identity being x^p. In this case the inverse behaves just like the group inverse.

Lemma 6.2.2. *The inverse semigroup direct product $S = C_{p_1} \times C_{p_2} \times ... \times C_{p_n}$ is presented by the inverse semigroup Coxeter presentation*

$$\langle x_1, x_2, ..., x_n | x_i^{p_i+1} = x_1,\ x_j x_k = x_k x_j\ (1 \leq i, j, k \leq n)\rangle.$$

PROOF: We know that S is presented by

$$\langle x_1, x_2, ..., x_n | x_i^{p_i+1} = x_1, x_j x_k = x_k x_j, x_j {x_k}^{-1} = {x_k}^{-1} x_j (1 \leq i, j, k \leq n, j \neq k)\rangle.$$

We need to eliminate the relations of the form $x_j {x_k}^{-1} = {x_k}^{-1} x_j$. This is done by noticing that ${x_k}^{-1} = {x_k}^{p_k - 1}$ and that $x_j {x_k}^{p_k-1} = {x_k}^{p_k-1} x_j$. ■

It follows that in a Coxeter inverse semigroup, $S = (X \cup X^{-1})^*/\tau$ that $(xx^{-1})\tau = (x^{-1}x)\tau$ for $x \in X \cup X^{-1}$. It follows that S is generated by X as a semigroup.

Lemma 6.2.3. *Let S be an inverse semigroup with $x, y \in S$ such that $x^{p_1+1} = x$, $y^{p_2+1} = y$ and $(xy)^q = x^{p_1}$, then*

$$xx^{-1}yy^{-1} = xx^{-1}$$

and

$$xyy^{-1}x^{-1} = yxx^{-1}y^{-1} = xx^{-1}$$

PROOF: By Lemma 6.2.2 $xx^{-1} = x^{-1}x = x^{p_1}$ and $yy^{-1} = y^{-1}y = y^{p_2}$. Therefore

$$xx^{-1}yy^{-1} = (xy)^q yy^{-1} = (xy)^q y^{-1}y = (xy)^q = xx^{-1}.$$

Also

$$\begin{aligned} xyy^{-1}x^{-1} &= x(x^{-1}x)yy^{-1}x^{-1} \\ &= x(xy)^q yy^{-1}x^{-1} \\ &= x(xy)^{q-1}xyy^{-1}yx^{-1} \\ &= x(xy)^q x^{-1} \\ &= xx^{-1}xx^{-1} \\ &= xx^{-1}. \end{aligned}$$

As $H_{xx^{-1}}$ is a group and $xx^{-1}\mathcal{R}xx^{-1}y^{-1}$ and $xx^{-1}\mathcal{L}xx^{-1}y^{-1}$ then $xx^{-1}y^{-1} = y^{-1}xx^{-1}$ and so

$$yxx^{-1}y^{-1} = yy^{-1}xx^{-1} = xx^{-1}.$$

■

Definition 6.2.4. A semigroup which is a semilattice of groups is *Clifford semigroup*.

The following theorem helps us identify Clifford semigroups. The proof is found in Petrich [18].

Theorem 6.2.5. *The following conditions on a semigroup S are equivalent.*

(i) *S is a Clifford semigroup.*

(ii) *S is regular and if e is an idempotent in S then $es = se$ for all $s \in S$.*

(iii) *S is an inverse semigroup and $ss^{-1} = s^{-1}s$ for all $s \in S$.*

Theorem 6.2.6. *All Coxeter inverse semigroups are Clifford semigroups.*

PROOF: Let S be a Coxeter inverse semigroup and let $u = x_1x_2...x_n \in S$ where $x_1, x_2, ..., x_n \in X$ (there is no need to include any elements of X^{-1} as they can be rewritten as elements of X using Lemma 6.2.2). Then

$$uu^{-1} = x_1...x_{n-1}x_n{x_n}^{-1}{x_{n-1}}^{-1}...{x_1}^{-1}$$

and there are four cases which can arise

- $x_{n-1} = x_n$. In which case

$$\begin{aligned} x_{n-1}x_n{x_n}^{-1}{x_{n-1}}^{-1} &= {x_n}^2{x_n}^{-2} \\ &= x_n{x_n}^{-1}x_n{x_n}^{-1} \\ &= x_{n-1}{x_{n-1}}^{-1}x_n{x_n}^{-1} \end{aligned}$$

- $x_{n-1}x_n = x_nx_{n-1}$. In which case

$$x_{n-1}x_n{x_n}^{-1}{x_{n-1}}^{-1} = x_{n-1}{x_{n-1}}^{-1}x_n{x_n}^{-1}.$$

- $(x_{n-1}x_n)^{q_{n-1n}} = {x_{n-1}}^{p_{n-1}}$. In which case by Lemma 6.2.3

$$x_{n-1}x_n{x_n}^{-1}{x_{n-1}}^{-1} = x_{n-1}{x_{n-1}}^{-1} = x_{n-1}{x_{n-1}}^{-1}x_n{x_n}^{-1}.$$

- $(x_nx_{n-1})^{q_{nn-1}} = {x_n}^{p_n}$. In which case by Lemma 6.2.3

$$x_{n-1}x_n{x_n}^{-1}{x_{n-1}}^{-1} = x_n{x_n}^{-1} = x_{n-1}{x_{n-1}}^{-1}x_n{x_n}^{-1}.$$

So in each case

$$uu^{-1} = x_1...x_{n-2}(x_{n-1}x_{n-1}{}^{-1}x_nx_n{}^{-1})x_{n-2}{}^{-1}...x_1{}^{-1},$$

and noticing that $(x_{n-1}x_{n-1}{}^{-1}x_nx_n{}^{-1}) = yy^{-1}$ for some $y \in S$ we may repeat this procedure until we arrive at:

$$uu^{-1} = x_1x_1{}^{-1}x_2x_2{}^{-1}...x_nx_n{}^{-1}.$$

As each x_i commutes with $x_i{}^{-1}$ then it is easy to see that the same procedure applies to $u^{-1}u$ and so $uu^{-1} = u^{-1}u$ and therefore S is a Clifford semigroup by Theorem 6.2.5 ■

EXAMPLE: Let P be the Coxeter presentation

$$\langle x, y | x^3 = x,\ y^5 = y,\ (xy)^2 = x^2 \rangle$$

Then the inverse semigroup presented by P has two $\mathcal{R}$-classes, R_y which is isomorphic to the cyclic group C_4 and R_x which is isomorphic to the dihedral group D_4. This can be seen by the fact that if we start with the linear graph Γ_y, then the only relation which contains y as a subword is $y^5 = y$. If we notice that in the word graph generated by applying the elementary $\mathcal{P}$-expansion corresponding to $y^5 = y$ there are no edges labelled by x and so there is no way that either of the other two relations can be applied and we are finished. On the other hand if we apply the elementary $\mathcal{P}$-expansion corresponding to $x^3 = x$ to the linear graph Γ_x then we obtain a word graph which contains a path labelled by x^2 and we may therefore apply the elementary P-expansion corresponding to $(xy)^3 = x^2$. At this point the word graph contains edges labelled by y and all three relations can be applied and by Theorem 6.0.2, R_x is isomorphic to the dihedral group presented by

$$\langle x, y | x^2 = y^4 = (xy)^2 = \epsilon \rangle.$$

6.3 Symmetric Presentations

This section looks at a type of presentation examined in the paper *On a Class of Semigroups with Symmetric Presentations* by Campbell, Robertson and Thomas

A *symmetric presentation* is a semigroup presentation of the form

$$\Sigma(m,n) = \langle x_1, x_2, ..., x_n | {x_i}^{m+1} = x_i (1 \le i \le n), x_i x_j^2 = x_j x_i^2 (1 \le i < j \le n) \rangle.$$

The semigroup presented by $\Sigma(m,n)$ is denoted $S(m,n)$, the group presented by $\Sigma(m,n)$ is denoted by $G(m,n)$ and the inverse semigroup presented by $\Sigma(m,n)$ is denoted by $IS(m,n)$.

$\Sigma(m,n)$ is called a symmetric presentation because any permutation of the generators produces a permutation of the relations. They also have the property that for every relation $u = v$, the generators involved in u are exactly the same as the generators involved in v so that any two words in X^* representing the same element in the semigroup $S(m,n)$ will involve precisely the same generators. $S(m,n)$ is therefore a semilattice of semigroups, where the semilattice is the Boolean lattice of subsets of the set of generators under reverse inclusion, with the empty set removed.

When m is odd then $S(m,n)$ is a semilattice of groups and therefore an inverse semigroup (in particular a Clifford semigroup). In this case the inverse semigroup $IS(m,n)$ presented by the $\Sigma(m,n)$ is isomorphic to the semigroup presented by $\Sigma(m,n)$ because in a finite Clifford semigroup S, if $u \in S$ then $u^{-1} = u^p$ for some $p \ge 0$ and we can thus eliminate the inverses (cf Theorem 6.2.6). Thus insofar as we know anything about $S(m,n)$ then we can say the same about $IS(m,n)$. This case is useful for testing of the enumerator algorithm, because we already know what the results should be.

If m is even and greater than 6 then $S(m,n)$ is infinite, I shall look at some examples of $\Sigma(2,n)$ and $\Sigma(4,n)$.

EXAMPLE: The semigroup $S(2,2) = \langle x, y | x^3 = x,\ y^3 = y,\ xy^2 = yx^2 \rangle$ has five $\mathcal{L}$-classes each of which is isomorphic to the cyclic group C_2. The elements are given in the following table:

$\mathcal{D}$-class	$\mathcal{L}$-class	Elements
D_x	L_x	x, x^2
D_y	L_y	y, y^2
D_{xy}	L_{xy}	xy, x^2y
D_{xy}	L_{yx}	yx, y^2x
D_{x^2yx}	L_{x^2yx}	x^2yx, x^2, y^2

Clearly $S(2,2)$ is not an inverse semigroup as D_{xy} contains two $\mathcal{L}$-classes but only one $\mathcal{R}$-class. We can see that $xy\mathcal{R}yx$ as $(xy)yx = (xy^2)x = yx^3 = yx$ and $(yx)xy = (yx^2)y = xy^3 = xy$. Also xy and yx are both idempotent because

$$(xy)^2 = xyxy = xyxy^3 = xy^2x^2y = yx^4y = yx^2y = xy^3 = xy$$

and in the same way $(yx)^2 = yx$.

Notice that in $IS(2,2)$, we have $(xy)^2 = xy$ and $(yx)^2 = yx$ as before, however by the commutativity of idempotents

$$xy = (yx)(xy) = (xy)(yx) = yx$$

and so we could say that Greens classes L_{xy} and L_{yx} in $S(2,2)$ are "identified with each other in $IS(2,2)$."

The enumeration of $IS(2,2)$ gives us the semilattice of cyclic groups:

$\mathcal{D}$-class	Elements
D_x	x, x^2
D_y	y, y^2
D_{xy}	xy, x^2y

EXAMPLE: The semigroup

$$S(4,3) = \langle x, y, z | x^5 = x,\ y^5 = y, z^5 = z,\ xy^2 = yx^2,\ xz^2 = zx^2,\ yz^2 = zy^2 \rangle$$

has 25 $\mathcal{L}$-classes which include

- L_x, L_y and L_z which are cyclic groups of order 4.
- $L_{xy}, L_{yx}, L_{xz}, L_{zx}, L_{yz}, L_{zy}$ - six groups of order 20 with $L_{xy}\mathcal{R}L_{yx}$, $L_{xz}\mathcal{R}L_{zx}$ and $L_{yz}\mathcal{R}L_{zy}$.
- 15 $\mathcal{L}$-classes of similar type to $L_{x(yz)^2}$ each with 84 elements.
- $L_{(x^2y^2z^2)^2} \cong G(4,3)$ which contains 100 elements.

By contrast $IS(4,3)$ contains

- L_x, L_y and L_Z which are cyclic groups of order 4.
- $L_{xy} = L_{yx}$, $L_{xz} = L_{zx}$, $L_{yz} = L_{zy}$ which are groups of order 20.
- $L_{xyz} \cong G(4,3)$ which is a group of order 100.

6.4 Free Products Involving a Semilattice

Firstly I shall define free products for inverse semigroups.

Definition 6.4.1. Given two inverse semigroups (inverse monoids) S and T with presentations $\langle X|U\rangle$ and $\langle Y|V\rangle$ so that $(X \cup X^{-1}) \cap (Y \cup Y^{-1}) = \emptyset$ then the *inverse semigroup free product* (*inverse monoid free product*) is the inverse semigroup (inverse monoid) $S * T$ which is presented as an inverse semigroup by $\langle X \cup Y|U \cup V\rangle$.

Inverse semigroup free products are, of course, defined in a similar way to group and semigroup free products. This does not make them semigroup or group free products. If, for example, S is an inverse semigroup presented by $\langle X|U\rangle$ and T is an inverse semigroup presented by $\langle Y|V\rangle$ then the inverse semigroup free product $S*T$ will have, for example, implicit relations of the form $(xx^{-1}yy^{-1}, yy^{-1}xx^{-1})$ for $x \in (X \cup X^{-1})^*$ and $y \in (Y \cup Y^{-1})^*$ whereas the semigroup free product will not.

The inverse semigroup free product between two inverse semigroups S and T will produce an infinite inverse semigroup unless S is finite and T is a semilattice (or vice versa). This is because if both S and T contain the non-idempotent elements $s \in S$ and $t \in T$ then $(st)^k \in S * T$ are distinct for all k.

I shall characterise these inverse semigroup with the following two lemmas.

Lemma 6.4.2. *Let S be an inverse semigroup and let L be a semilattice such that $S \cap L = \emptyset$. The the set of idempotents of $S * L$ is a semilattice generated by*

$$\Lambda = L \cup \{ses^{-1}|s \in S,\ e \in L\} \cup \{ss^{-1}|s \in S\}$$

PROOF: Now

$$(ses^{-1})^2 = se(s^{-1}s)es^{-1} = ss^{-1}se^2s^{-1} = ses^{-1}$$

and so by commutativity of idempotents, Λ generates a semilattice. Also if $t \in S$ and $f \in L$ then as the product is free ses^{-1} is distinct from tft^{-1} if either $u \neq v$ or $e \neq f$.

Let $s = s_1e_1s_2e_2...e_{n-1}s_n$ where $s_i \in S$ for $1 \leq i \leq n$ and where $e_i \in L$ for $1 \leq i \leq n-1$ and where each $|s_i| > 0$ and $n > 0$. Now for any $1 \leq i \leq n-1$

and if we define $t_i = s_1 s_2 ... s_i$ then

$$\begin{aligned} t_i e_i t_i{}^{-1} t_{i+1} e_{i+1} t_{i+1}{}^{-1} &= (s_1...s_i)e_i((s_1...s_i)^{-1}(s_1...s_i))s_{i+1}e_{i+1}(s_1...s_{i+1}) \\ &= (s_1...s_i)(s_1...s_i)^{-1}(s_1...s_i)e_i s_{i+1} e_{i+1}(s_1...s_{i+1}) \\ &= t_i e_i s_{i+1} e_{i+1} t_{i+1} \end{aligned}$$

and

$$\begin{aligned} t_{n-1}{}^{-1} t_n t_n{}^{-1} &= t_{n-1}{}^{-1} t_{n-1} s_n s_n{}^{-1} t_{n-1}{}^{-1} \\ &= s_n s_n{}^{-1} t_{n-1}{}^{-1} t_{n-1} t_{n-1}{}^{-1} \\ &= s_n t_n{}^{-1} \end{aligned}$$

and therefore

$$\begin{aligned} &(t_n t_n{}^{-1})(t_1 e_1 t_1{}^{-1})(t_2 e_2 t_2{}^{-1})...(t_{n-1} e_{n-1} t_{n-1}{}^{-1})(t_n t_n{}^{-1}) \\ &= t_1 e_1 s_2 e_2 ... s_{n-2} e_{n-2}(s_{n-1} e_{n-1} t_{n-1}{}^{-1})(t_n t_n{}^{-1}) \\ &= s_1 e_1 s_2 e_2 ... s_{n-2} e_{n-2}(s_{n-1} e_{n-1} s_n t_n{}^{-1}). \end{aligned}$$

Now for $2 \leq i \leq n$

$$\begin{aligned} e_{i-1} s_i ... s_n t_n{}^{-1} &= (e_{i-1})^2 (s_i ... s_n s_n{}^{-1} ... s_i{}^{-1}) s_{i-1}{}^{-1} ... s_1{}^{-1} \\ &= e_{i-1}(s_i ... s_n s_n{}^{-1} ... s_1{}^{-1}) e_{i-1} s_{i-1}{}^{-1} ... s_1{}^{-1} \end{aligned}$$

and combining these two results we get

$$\begin{aligned} &(t_n t_n{}^{-1})(t_1 e_1 t_1{}^{-1})(t_2 e_2 t_2{}^{-1})...(t_{n-1} e_{n-1} t_{n-1}{}^{-1})(t_n t_n{}^{-1}) \\ &= (s_1 e_1 ... e_{n-1} s_n)(s_n{}^{-1} e_{n-1} ... e_1 s_1{}^{-1}) \end{aligned}$$

and we need only notice that the right hand side is any idempotent in $S * L \setminus (L \cup S)$ and the left hand side is a product of elements of

$$\{ses^{-1} | s \in S,\ e \in L\} \cup \{ss^{-1} | s \in S\}.$$

■

Lemma 6.4.3. *Let S be an inverse semigroup and let L be a semilattice then for any $s_1, s_2, ..., s_n \in S$ and $e_1, e_2, ..., e_{n-1} \in L$ then $S\Gamma_S(s_1 s_2 ... s_n)$ is embedded in $S\Gamma_{S*L}(s_1 e_1 s_2 e_2 ... e_{n-1} s_n)$ and $|V(S\Gamma_S(s_1 s_2 ... s_n))| = |V(S\Gamma_{S*L}(s_1 e_1 s_2 e_2 ... e_{n-1} s_n))|$.*

PROOF: As the product of S and L is free then we know that each of the e_i label a $(s_1 e_1 ... e_{i-1} s_i, s_1 e_2 ... e_{i-1} s_i)$-walk and there are no relations which we can use to expand this. Therefore e_i acts as an identity on this vertex and there are no other differences to $S\Gamma_S(s_1 ... s_n)$. ■

It is easiest to see what is going on in the special case of $G * L$ where G is a group and L is a semilattice containing only one element. Here are a couple of examples.

EXAMPLE: Let G be the cyclic group presented by the inverse semigroup presentation $\langle x|x^4 = x\rangle$. The table below lists all the elements of the inverse semigroup presented by $\langle x, e|x^4 = x, e^2 = e\rangle$.

$\mathcal{D}$-class	$\mathcal{R}$-class	Elements
D_e	R_e	e
D_x	R_x	x, x^2, x^3
D_{xe}	$R_{x^3ex^3}$	x^3ex^3, x^3ex, x^3ex^2
D_{xe}	R_{xex^2}	xex^3, xex, xex^2
D_{xe}	R_{x^2ex}	x^2ex^3, x^2ex, x^2ex^2
D_{exe}	$R_{ex^3ex^3}$	$ex^3ex^3, ex^3ex, ex^3ex^2$
D_{exe}	R_{exex^2}	$exex^3, exex, exex^2$
D_{exe}	R_{ex^2ex}	$ex^2ex^3, ex^2ex, ex^2ex^2$
D_{exexe}	R_{exexex}	$exexex^2, exexe, exexex$

EXAMPLE: It is interesting to examine the difference between the semigroup S presented by $\langle x, e|x^3 = x,\ e^2 = e\rangle$ and the inverse semigroup S' presented by the same presentation. It is not difficult to see that the former contains all the (distinct) elements of the form $(xe)^i$ for any $i > 0$ and is thus infinite. On the other hand in the inverse semigroup S', by commutativity of idempotents

$$(xex)^2 = xex^2ex = x^3ex = xex$$

and so $(xe)^{2i}$ can be rewritten as $((xex)e)^i$ and by commutativity of idempotents $(xe)^{2i} = (xex)^ie^i = xexe$. Similarly $(xe)^{2i+1} = (xex)exe = exex^2e = exe$. From a presentation theory perspective we can reduce words to canonical forms using not only relations but by recognising idempotents and allowing them to commute.

If G is a group and L a semilattice, then thinking about the Schützenberger graphs of $G * L$, we have a semilattice of groups all of which are isomorphic to G. For some $u \in G * L$ with

$$uu^{-1} = (g_1e_1g_1{}^{-1})(g_2e_2g_2{}^{-1})...(g_ne_{n-1}g_n{}^{-1})$$

where $g_1, g_2, ..., g_n \in G$ and $e_1, e_2, ..., e_{n-1} \in L$ then $S\Gamma(u)$ is the Cayley graph of G with each vertex labelled by u_i "coloured" by the idempotent e_i. That is each vertex u_i has a (u_i, u_i)-walk of length 1 labelled by e_i attached to it. Every other vertex is "uncoloured".

We can construct new Schützenberger graphs from known Schützenberger graphs. Suppose $vv^{-1} = uu^{-1}(g_{n+1}e_n g_{n+1}{}^{-1})$ (with $g_{n+1} \in G$ and $e_n \in L$). If $g_{n+1} = g_i$ for some $1 \leq i \leq n$, then the vertex g_i is "recoloured" by $e_i e_n$ in $S\Gamma(v)$ otherwise $S\Gamma(v)$ is identical to $S\Gamma(u)$. If on the other hand $g_{n+1} \neq g_i$ for any $1 \leq i \leq n$, then the vertex g_{n+1} is coloured by e_n in $S\Gamma(v)$ and otherwise $S\Gamma(v)$ is identical to $S\Gamma(u)$.

It is worth noting that if both G and L are finite then $G * L$ has a minimum idempotent, ω which is the product of all elements of the form geg^{-1} for $g \in G$ and $e \in L$. In this case $S\Gamma(\omega)$ is the Cayley graph of G with each vertex coloured by the least element in L.

We have the following theorem.

Theorem 6.4.4. *If G is a finite group and L is a finite semilattice such that $G \cap L = \emptyset$, then*

$$|G * L| = |G| * (|L| + 1)^{|G|} + |L|.$$

PROOF: Each Schützenberger graph in $G * L$ is either a single vertex labelled by an element of L or the Cayley graph or G with each vertex coloured by an element of L or not coloured at all. We therefore have $|L| + 1$ options for each vertex and so there are $(|L| + 1)^{|G|}$ Schützenberger graphs of order $|G|$ and $|L|$ Schützenberger graphs of order 1 in $G * L$. ■

For the more complex case of $S * L$ where S is a finite inverse semigroup and L a semilattice we have.

Theorem 6.4.5. *If S is a finite inverse semigroup with a set of idempotents E and L is a finite semilattice such that $S \cap L = \emptyset$, then*

$$|S * L| = \Sigma_{e \in E}(|R_e| * (|L| + 1)^{|R_e|}) + |L|.$$

PROOF: The reasoning is the same as Theorem 6.4.4 except that we apply the same logic for each $\mathcal{R}$-class in S as we did to G and then sum the results. ■

6.5 On Inverse Semigroups with infinite $\mathcal{R}$-classes

Although at first sight the inverse monoid enumerator is quite awkward because it enumerates $\mathcal{R}$-classes separately, there is however an advantage to this in that it is capable of enumerating single finite $\mathcal{R}$-classes in infinite inverse semigroups. Unlike the previous examples we have looked at, there are many examples of inverse semigroup presentations of inverse semigroups of this type which are not embedded in the semigroup given by the same presentation.

EXAMPLE: The most obvious example of an infinite inverse monoid in which every $\mathcal{R}$-class in finite is $\mathbf{F}_{\mathcal{IM}}(X)$. In this case the algorithm simply gives a table which corresponds to the word tree of the word which generates the $\mathcal{R}$-class.

EXAMPLE: Let S be the inverse semigroup presented by $\langle x, y | xy = (xy)^2 \rangle$. Now let $u \in \mathbf{F}_{\mathcal{IS}}(x, y)$ and suppose that the word tree T_{xy} cannot be embedded in T_u then $S\Gamma_S(u) = T_u$ as there is no way to apply any elementary $\mathcal{P}$-expansions. Otherwise suppose there is a (v, vxy)-walk labelled by xy in T_u, then $vxy = vxyxy$ and as $v\mathcal{R}vxy$ then $vxyy^{-1}x^{-1} = vxy$ however $v\mathcal{R}vxyy^{-1}x^{-1}$ and so $v = vxyy^{-1}x^{-1} = vxy$ and so xy labels a (v, v)-walk in $S\Gamma_S(u)$. For example if $u = xyy^{-1}x^{-1}y$ then T_u is the following tree.

$$\begin{array}{cccc} & \gamma_3 & & \gamma_4 \\ & \uparrow_y & & \uparrow_y \\ \rightarrow & \gamma_1 & \rightarrow_x & \gamma_2 \end{array}$$

where $\gamma_1 = xyy^{-1}x^{-1}yy^{-1}$, $\gamma_2 = yy^{-1}xyy^{-1}$, $\gamma_3 = xyy^{-1}x^{-1}y$ and $\gamma_4 = yy^{-1}xy$ while $S\Gamma_S(u)$ is the following graph where $\gamma_4 := \gamma_1$.

$$\begin{array}{cccc} & \gamma_3 & & \\ & \uparrow_y & & \\ \rightarrow & \gamma_1 & \rightarrow_x & \gamma_2 \\ & & \leftarrow_y & \end{array}$$

EXAMPLE: Let S be the inverse semigroup presented by

$$P = \langle x, y | x^3 = x^2, y^3 = y^2, xy = yx \rangle.$$

At a glance S seems to be finite as both the group and the semigroup defined by P are finite and commutative, indeed the group is trivial. The inverse semigroup, is however infinite because $xy^{-1} \neq y^{-1}x$ and it turns out that each $(xy^{-1})^i$ is distinct for all $i > 0$. As with the above example we can, however, readily enumerate the $\mathcal{R}$-classes of S. For example $S\Gamma(xyx) = S\Gamma(x^2y)$ looks like:

$$\begin{tikzcd} {} \arrow[r] & x^2yy^{-1} \arrow[loop above, "x"] \arrow[r, "y"] & x^2y \arrow[loop above, "x"] \end{tikzcd}$$

Whereas $S\Gamma(xy^{-1}x)$ is the word tree

$$\begin{tikzcd} {} \arrow[r] & \gamma_1 \arrow[r, "x"] & \gamma_2 & \\ & & \gamma_3 \arrow[u, "y"] \arrow[r, "x"] & \gamma_4 \end{tikzcd}$$

6.6 On Inverse Semigroups with an infinite $\mathcal{R}$-class

In this section I look at some of the examples I looked at with Allessandra Cherubini and Brunnetto Piochi. I use the technique for enumerating R_u/ζ and R_u/H that I developed (see **Section 5.5**).

EXAMPLE: Let the inverse semigroup S be presented by

$$\langle x, y, e | xx^{-1} = x^{-1}x,\ y^3 = y,\ e^2 = e,\ xy = yx,\ xe = ex \rangle.$$

Now $\langle x \rangle$ generates a free group, $\langle y \rangle$ generates a cyclic group of order 2 and $\langle x, y \rangle$ is the direct product of the two groups. Now $eyx\mathcal{R}eyxx^{-1}\mathcal{R}eyx^i$ for all non-zero values of i and similarly $xey\mathcal{L}x^{-1}xey\mathcal{L}x^iey$ but $eyx = xey$ and so H_{eyx} contains an isomorphic copy of the free group $\langle x \rangle$. The $R_{eyy^{-1}xx^{-1}}/H$ enumerator will therefore find a right quotient generated by $(eyy^{-1}x, eyy^{-1}xx^{-1})$. The word graph for $S\Gamma(eyx)/H$ is:

$$\begin{tikzcd} {} \arrow[r] & \gamma_1 \arrow[loop above, "{x,e}"] \arrow[r, leftrightarrow, "y"] & \gamma_2 \arrow[loop above, "x"] \end{tikzcd}$$

Similarly the word graph for $S\Gamma(yex)/H$ is:

$$\begin{tikzcd} {} \arrow[r] & \gamma_1 \arrow[loop above, "x"] \arrow[r, leftrightarrow, "y"] & \gamma_2 \arrow[loop above, "{x,e}"] \end{tikzcd}$$

Other than these two $\mathcal{R}$-classes there are $R_x = \langle x \rangle$, $R_y = \langle y \rangle$, $R_e = \{e\}$, $R_{ex} \cong \langle x \rangle$, $R_{ey} = \{ey, ey^2\}$ which is $\mathcal{L}$ related to $R_{ye} = \{ye, yey\}$, R_{xy} and R_{eyex} with the last two being isomorphic to $\langle x, y\rangle$. It is interesting to note that $S\Gamma(ey)$ is almost identical to $S\Gamma(eyx)/H$ with the only difference being the lack of edges labelled by x. Similarly $S\Gamma(ye)$ is almost identical to $S\Gamma(yex)/H$.

Definition 6.6.1. An inverse monoid presentation where the relations are made up of idempotent relations (see Definition 6.1.3) is called an *idempotent presentation*. An inverse semigroup (inverse monoid) presented by an idempotent presentation is called an *idempotent inverse semigroup* (*idempotent inverse monoid*).

Inverse semigroups and inverse monoids defined by idempotent presentations are quite unusual, especially if we note that groups and semigroups with such relations are free. Indeed it is easy to see that all Schützenberger graphs in an inverse semigroup (inverse monoid) defined by the idempotent presentation $\langle X|U\rangle$ can be embedded in the Cayley graph of $\mathbf{F}_{\mathcal{G}}(X)$.

Lemma 6.6.2. *If S is an idempotent inverse semigroup then for any idempotent $e \in S$, H_e is a free group.*

PROOF: Suppose that H_e is generated by $Y \subseteq S$. We know that H_e is a group by Corollary 2.1.7 and so each of the relations in the presentation for S is trivial on Y^*. ■

NOTE: Note in the lemma above that H_e could be a free group with zero generators, in which case $H_e = \{e\}$.

EXAMPLE: Let S be an idempotent inverse semigroup presented by

$$\langle x, y|xx^{-1} = y^{-1}y, x^{-1}x = y^2y^{-2}\rangle.$$

It turns out that every $\mathcal{R}$-class in S contains an infinite number of $\mathcal{H}$-classes, this demonstrates a failing in the potentials of the inverse monoid enumerator. However it is actually very easy to work out the structure of each of the $\mathcal{R}$-classes by hand.

Now $S\Gamma(x)$ will certainly contain the word subgraph

$$\rightarrow xx^{-1} \rightarrow_x x$$

Noticing that the path (xx^{-1}, x, xx^{-1}) is labelled by xx^{-1} and that the path (x, xx^{-1}, x) is labelled by $x^{-1}x$ then we can immediately perform two elementary $\mathcal{P}$-expansions to get

$$
\begin{array}{ccccc}
 & & & & xy^2 \\
 & & & & \uparrow_y \\
 & & & & xy \\
 & & & & \uparrow_y \\
\rightarrow & xx^{-1} & & \rightarrow_x & x \\
 & \uparrow_y & & & \\
 & xx^{-1}y^{-1} & & &
\end{array}
$$

I shall call this the *base graph.* If we now notice that the paths (xy, x, xy) and (xy^2, xy, xy^2) are labelled by $y^{-1}y$ and noting that although the path $(xx^{-1}, xx^{-1}y^{-1})$ is labelled by $y^{-1}y$ it is already possible to trace a walk labelled by xx^{-1} starting at the vertex xx^{-1}, we can perform another two elementary $\mathcal{P}$-expansions to get

$$
\begin{array}{ccccccc}
 & & & & xy^2 & \rightarrow_x & xy^2x \\
 & & & & \uparrow_y & & \\
 & & & & xy & \rightarrow_x & xyx \\
 & & & & \uparrow_y & & \\
\rightarrow & xx^{-1} & & \rightarrow_x & x & & \\
 & \uparrow_y & & & & & \\
 & xx^{-1}y^{-1} & & & & &
\end{array}
$$

At this point we notice that the whole procedure can be repeated as we have another two walks labelled by xx^{-1}, namely (xy, xyx, xy) and (xy^2, xy^2x, xy^2). In essence, after we have constructed the base graph the following expansions see the attachment to the vertices xy and xy^2 two subgraphs of the form

$$
\begin{array}{ccc}
 & & \gamma_4 \\
 & & \uparrow_y \\
 & & \gamma_3 \\
 & & \uparrow_y \\
\gamma_1 & \rightarrow_x & \gamma_2
\end{array}
$$

I shall call the above graph the *repeated graph.*

Likewise $S\Gamma(y)$ has a base graph which is isomorphic to the base graph for $S\Gamma(x)$.

$$\begin{array}{ccccc}
 & & & yxy^2 \\
 & & & \uparrow_y \\
 & & & yxy \\
 & & & \uparrow_y \\
 & y & \rightarrow_x & yx \\
 & \uparrow_y & & \\
\rightarrow & yy^{-1} & &
\end{array}$$

Again the same repeated graph is attached, this time to the vertices yxy and yxy^2. If we now look at $S\Gamma(x^2)$ we have a base graph of

$$\begin{array}{cccccc}
 & & & x^2x^{-1}y^2 & & x^2y^2 \\
 & & & \uparrow_y & & \uparrow_y \\
 & & & x^2x^{-1}y & & x^2y \\
 & & & \uparrow_y & & \uparrow_y \\
\rightarrow & x^2x^{-2} & \rightarrow_x & x^2x^{-1} & \rightarrow_x & x^2 \\
 & \uparrow_y & & \uparrow_y & & \\
 & x^2x^{-2}y^{-1} & & x^2x^{-1}y^{-1} & &
\end{array}$$

Here we simply attach the repeated graph to the vertices $x^2x^{-1}y$, $x^2x^{-1}y^2$, x^2y and x^2y^2.

As we can see all these Schützenberger graphs contain what I loosely term a repeated graph, a base and a *tail* on the base. Where the base contains a certain number of copies of the repeated graph with a tail. In $S\Gamma(x)$ the tail is the subgraph

$$\begin{array}{c}
xx^{-1} \\
\uparrow_y \\
xx^{-1}y^{-1}
\end{array}$$

while the base for $S\Gamma(x^2)$ is two copies of the base for $S\Gamma(x)$.

All this is perhaps leading to a more sophisticated technique for enumerating $\mathcal{R}$-classes in idempotent inverse semigroups, where instead of factoring out right congruences, repeated graphs are factored out. If there is such a method then the key to it is in recognising the boundaries to the base graph.

Imagine a *base graph enumerator* which operates in a similar manner to the $\mathcal{R}$-class enumerator. If there is a $v \in R_u$ such that $uu^{-1}v \leq vuu^{-1}$ then after tracing a path labelled by v in $S\Gamma(u)$ we can immediately trace another copy of $S\Gamma(u)$ starting from the vertex labelled by $uu^{-1}v$. As soon as such a v is found then the base graph enumerator labels that vertex as a *boundary* and no further vertices are defined adjoining this vertex. Assuming that there is a terminating process where all the boundaries are found then imagining that another set of base graphs are adjoined by an elementary $\mathcal{P}$-expansions, then there will be a certain "overlap" between the original base graph and the adjoined set of base graphs. This overlap is the *tail*. The new adjoined base graph without the tail is the *repeated graph* which is continually adjoined to the previous repeated graph or in the first instance the base graph. The algorithm should return the base graph, its boundaries and the repeated graph.

EXAMPLE: The easiest example of an idempotent inverse semigroup which contains $\mathcal{R}$-classes with infinite $\mathcal{H}$-classes is the bicyclic monoid, B_x presented by $\langle x|xx^{-1} = \epsilon\rangle$. Given the idempotent $x^{-m}x^m \in B_x$, then the base graph for $S\Gamma(x^{-m}x^m)(= S\Gamma(x^{-m}))$ is simply the linear graph $\Gamma_{x^{-m}}$.

$$x^{-m} \rightarrow_x x^{-m}x \rightarrow_x \quad . \quad . \quad . \quad \rightarrow_x \underset{\uparrow}{x^{-m}x^m}$$

The first boundary that is found is $x^{-m}x$ as

$$(x^{-m}x^m)x = x^{-m}x^{m+1} \leq x^{-m+1}x^m = x(x^{-m}x^m).$$

This means that the repeated graph is

$$\gamma_1 \rightarrow_x \gamma_2$$

while the tail is

$$x^{-m}x \rightarrow_x x^{-m}x^2 \rightarrow_x \quad . \quad . \quad . \quad \rightarrow_x x^{-m}x^m$$

The most striking problem with this sort of procedure is that there are redundant boundaries at each of the vertices labelled by $x^{-m}x^i$ where $2 \leq i \leq m$. It does, however, look like it is possible to create a meaningful algorithm for finding the structure for these types of inverse semigroup.

Chapter 7

On the Automaticity of the Free Inverse Semigroup

The chapter constitutes a paper that I wrote with Andrew Solomon. It is related to rest of the thesis in that it looks at the structure of free inverse semigroups although it does not involve any references to coset enumeration.

7.1 Introduction

Automatic groups are widely studied and are the subject of a major book [5]. In [4] the notion of automaticity is extended to semigroups. The motivation of the present work is to determine whether free inverse semigroups are automatic. In the process of showing that they are not, we demonstrate that for these purposes, it is the property of having a regular set of unique normal forms that is of interest, a property considered in the context of groups by Gilman [7]. Connections with growth are exploited to prove the main theorem, and we also discuss decidability and the word problem.

We proceed now to recall some relevant definitions and notation. For any set X, X^* denotes the set of all words in the elements of X including the empty word ϵ, while X^+ denotes the set of all such words of length at least 1. We refer to the words of length 1 as *letters*. When X^* (respectively X^+) is considered along with the associative binary operation of concatenation, it is referred to as the *free monoid* (respectively *free semigroup*) on the set X, and has the universal properties one would expect. A *language* over X is a subset of X^*.

Let S be a semigroup and X a set of generators with natural homomorphism $\phi : X^+ \to S$. If $L \subseteq X^+$ is any language such that the restriction of ϕ to L is surjective, say that L is a *language of normal forms for S over X*. If in addition the restricition of ϕ to L is injective, say that L is a language of *unique* normal forms for S over X.

The fact that regular languages are precisely the sets accepted by finite state machines has passed into folklore and we use it freely without comment. For details see [8].

We set out some well known facts about regular languages for later reference.

Theorem 7.1.1. *Suppose X and Y are finite sets. Then*

(i) if $K \subseteq Y^*$ is a regular language and $\phi : Y^* \to X^*$ is a monoid homomorphism, then $\phi(K)$ is a regular language over X;

(ii) if $K, L \subseteq Y^*$ are regular languages, then so are $K \cup L$, $K \cap L$, $K \setminus L$, KL, K^* and K^+.

For convenience, we shall refer to a semigroup with a regular set of unique normal forms as a *rational* semigroup. We will see that in contrast with automaticity in semigroups, the property of being rational is independent of the choice of generating set. (This dependence of automaticity on choice of generating set is peculiar to semigroups, while an automatic monoid will have an automatic structure for any finite generating set – see [6] for details.)

7.1.1 Rational semigroups and automaticity

Although the developments in this paper do not depend on the definition of automaticity, we sketch it here by way of background and refer the interested reader to [4] for details. Let S be a semigroup with generating set A and natural homomorphism $\phi : A^+ \to S$. An *automatic structure* for S consists of a regular language $L \subseteq A^+$ of normal forms for S such that (roughly speaking) checking whether two words of L are equal or differ by a factor of a generator can be done by a finite state machine. Any semigroup with an automatic structure over some generating set is called an *automatic semigroup*.

An immediate consequence of [4, Corollary 5.6] is that

Lemma 7.1.2. *Any automatic semigroup is a rational semigroup.*

While an automatic semigroup may have an automatic structure over one generating set and not another, we show that the definition of a rational semigroup is independent of the choice of generating sets.

Lemma 7.1.3. *If a semigroup has a regular language of unique normal forms over some finite generating set, then it has a regular language of unique normal forms over every finite generating set.*

PROOF: Let L be a regular language of unique normal forms for a semigroup S over some finite generating set Y. Let $\psi_Y : Y^+ \to S$ be the natural homomorphism. Let X be some other generating set for S with natural homomorphism $\psi_X : X^+ \to S$. Then there is a function $\phi : Y \to X^+$ expressing every generator $y \in Y$ as a product of generators in X such that $\psi_X\phi(y) = \psi_Y(y)$. Extend ϕ to a homomorphism. By Theorem 7.1.1, $\phi(L)$ is a regular language. By definition of ϕ, $\psi_X\phi = \psi_Y$, so that since ψ_Y restricted to L is a bijection, so is ψ_X restricted to $\phi(L)$, proving that $\phi(L)$ is a regular language of unique normal forms for S over X. ■

On the other hand, the stronger definition of an automatic semigroup gives rise to a number of interesting properties, most significantly

Theorem 7.1.4 (2, Corollary 3.7). *If S is an automatic semigroup, we can solve the word problem for S in time quadratic in the length of the words.*

7.1.2 Rational semigroups and decidability

We show here that for rather general reasons, rational semigroups have a solvable (recursive) word problem and that the property of being rational is therefore Markov. It has been shown that for finitely presented semigroups [14], [15], groups [1], [19] and inverse semigroups [32], Markov properties are undecidable. For general background on computability, the reader is referred to [8].

Recall that a set is *recursively enumerable* if there is an algorithm to list its elements. We shall say that the word problem of a semigroup is *recursively enumerable* if there is an algorithm which lists all pairs of words in the generators which represent equal elements of the semigroup. It is a simple observation that a finitely presented semigroup has recursively enumerable word problem. For a finitely presented semigroup S and word w in the generators of S, denote by S_w the recursively enumerable set of elements of S equal to w in S.

The word problem for a semigroup is *recursive* (or *solvable*) if there is an algorithm whose input is two words in the generators and which terminates with output 'yes' if they represent the same element of the semigroup and terminates with output 'no' otherwise.

Theorem 7.1.5. *Let S be a finitely presented semigroup. Then the word problem for S is solvable if and only if S has a recursively enumerable set of unique normal forms.*

PROOF: Let A be a generating set for S. The direct part is obvious. If a semigroup has solvable word problem, simply list the elements of A^+ in some order. As we arrive at a word which represents the same element of S as another word already in the list, don't emit it but skip over it to the next word in A^+. In this way we are able to obtain a list of unique normal forms for elements of S.

Conversely, suppose there is a recursively enumerable set L of unique normal forms for S. Given words $u, v \in A^*$ we decide equality in S as follows:

- Since S_u is a recursively enumerable set and L is recursively enumerable, their intersection is also recursively enumerable. By uniqueness, this intersection is a singleton which we denote w_u;
- Compute the unique normal form w_v of v in the same way;
- u and v represent the same element of S precisely when $w_u = w_v$.

■

Since a regular language is trivially a recursively enumerable set we have

Corollary 7.1.6. *Rational semigroups (and therefore their finitely generated subsemigroups) have solvable word problem.*

This result is well known for semigroups which are groups, see for instance [5, Section 2.1].

Reflecting on the rather general argument above, we consider it an interesting question to determine what properties a semigroup will enjoy when the word problem and the set of unique normal forms are in other computability classes. For example, if the word problem were solvable by a push-down automaton or the set of unique normal forms were a context-free language.

A Markov property of semigroups [groups, inverse semigroups] is a property $\mathcal{P}$ such that:

- $\mathcal{P}$ is preserved under isomorphism;
- there is a finitely presented semigroup [group, inverse semigroup] which has property $\mathcal{P}$;
- there is a finitely presented semigroup [group, inverse semigroup] which embeds in no semigroup [group, inverse semigroup] with property $\mathcal{P}$.

As mentioned at the beginning of this section, it has been shown that Markov properties of semigroups, groups and inverse semigroups are undecidable. Among Markov properties is the property of having solvable word problem. However it is known [32] that there are undecidable properties which are not Markov.

Theorem 7.1.7. *The property of being rational is Markov for semigroups, groups and inverse semigroups.*

PROOF: Since the following argument is completely generic, the reader may replace 'semigroup' with 'group' or 'inverse semigroup' throughout, simply noting that there are finitely presented semigroups S in each class which are automatic and other finitely presented semigroups T in each class which have insoluble word problem. For details the reader is referred to [32].

By Lemma 7.1.3 we know that the property of being rational is preserved under isomorphism. Since every automatic semigroup is rational, there are certainly examples with this property. Let T be a finitely presented semigroup with insoluble word problem. Then by Corollary 7.1.6 T embeds in no semigroup which is rational. ∎

7.1.3 Closure operations on the class of rational semigroups

In this section we exhibit a number of operations under which the class of rational semigroups is closed. In the following discussion, if S is a semigroup, S^1 will denote the set S with an extra element 1 adjoined which is a multiplicative identity for every element of S^1, and S^0 will denote the set S with an extra element 0 adjoined which is a multiplicative zero for every element of S^0.

Theorem 7.1.8. *A finitely presented semigroup S is rational if and only if S^1 is rational.*

PROOF: Let S be a rational semigroup with regular language L of unique normal forms over generating set A. Let $B = A\dot{\cup}\{e\}$ be a generating set for S^1 where e maps to 1 under the natural homomorphism. Then L is a regular subset of B^+ and consequently so is $L' = L \cup \{e\}$. That L' is a regular set of unique normal forms for S^1 follows from the fact that there is no element of L which maps to $1 \in S^1$ under the natural homomorphism.

Conversely, suppose S^1 is rational. Then there is a set B of generators, a homomorphism $\phi : B^+ \to S^1$ and a regular language $L \subseteq B^+$ in bijection with S^1 under ϕ.

Firstly note that there is at least one letter $e \in B$ such that $\phi(e) = 1$, for otherwise 1 would be a product of non-identity elements of S, contradicting the defintion of S^1. Let $E \subseteq B$ be the set of all e such that $\phi(e) = 1$. Put $A = B \setminus E$ and define $\psi : B^* \to A^*$ by mapping all $e \in E$ to the empty word and fixing the other generators. Put w_1 equal to the preimage of 1 in L under ϕ, then the language $L \setminus \{w_1\}$ is regular and so is $\psi(L \setminus \{w_1\}) \subseteq A^*$. Since none of the elements of $L \setminus \{w_1\}$ are the empty word, nor composed entirely of letters of E, $\psi(L\setminus\{w_1\}) \subseteq A^+$. Defining $\gamma : A^+ \to S$ as the restriction of ϕ to A^+, we see that $\mathrm{Im}(\gamma) = \mathrm{Im}(\phi) \setminus \{1\} = S$, since for all $w \in B^+$, $\phi(w) = 1$ or $\phi(w) = \gamma\psi(w)$, so $\psi(L \setminus \{w_1\})$ is a set of normal forms. If $\gamma(u) = \gamma(v)$ for $u, v \in \psi(L \setminus \{w_1\})$, then $u = \psi(u')$ and $v = \psi(v')$ for some $u', v' \in L \setminus \{w_1\}$. Then

$$\phi(u') = \gamma\psi(u') = \gamma(u) = \gamma(v) = \gamma\psi(v') = \phi(v')$$

which shows that $u' = v'$ by injectivity of ϕ on $L \setminus \{w_1\}$. But then $u = v$ giving injectivity of γ on $\psi(L \setminus \{w_1\})$ as required. ■

A simpler argument gives

Theorem 7.1.9. *A finitely presented semigroup S is rational if and only if S^0 is rational.*

Theorem 7.1.10. *Let S be a rational semigroup and I an ideal of S such that S/I has no zero divisors. Then S/I is rational.*

PROOF: Suppose S has a regular language L of unique normal forms over some generating set A. Let $\natural_A : A^+ \to S$ be the natural homomorphism. Let $B = A \setminus \natural_A^{-1}(I) \dot{\cup} \{z\}$. Define $\natural_B : B \to S/I$ by

$$\natural_B(b) = \begin{cases} \natural_A(b) & \text{if } b \in A \setminus \natural_A^{-1}(I) \\ 0 & \text{if } b = z \end{cases}$$

and extend homomorphically. Under this mapping, B is clearly a generating set for S/I.

Let K be the regular language $(L \cap (B \setminus \{z\})^+) \cup \{z\}$ over B. To see that K is a set of normal forms for S/I, note that if $w \in L$ and $\natural_A(w) \in S \setminus I$, the fact that I is an ideal implies each letter of w is in B, so $w \in K$, whence the restriction of $\natural_B$ to K is onto.

Suppose $w_1, w_2 \in K$ and $\natural_B(w_1) = \natural_B(w_2) \in S \setminus I$, then $w_1, w_2 \in L$ so $w_1 = w_2$, by uniqueness in L. If $\natural_B(w) = 0$ then $w \notin K \setminus \{z\}$ since S/I has no zero divisors, therefore $w = z$. ∎

Theorem 7.1.11. *The free product of two semigroups is rational if and only if both factors are rational.*

PROOF: Let S and T be rational semigroups with regular languages of unique normal forms $K \subseteq A^+$ and $L \subseteq B^+$ respectively. The set $(LK)^+ \cup K(LK)^* \cup (LK)^*L \cup (KL)^+$ is again a regular language with a unique representative for each element of $S * T$ as required.

Conversely, suppose $S * T$ is a rational semigroup. The semigroups S^0 and T^0 are Rees quotients of $S * T$ without zero divisors, and are therefore rational by Theorem 7.1.10, and by Theorem 7.1.9, S and T are also rational. ∎

7.1.4 Growth and rational semigroups

We take the following development on the growth of functions from [30]. Consider the set of non-decreasing functions from $\mathbb{N} \to \mathbb{R}^+$. We define a preorder on this set by $f \preceq g$ if and only if there are positive natural numbers m and c such that for every $n \in \mathbb{N}$, $f(n) \leq cg(mn)$. Further define an equivalence relation $\sim$ by $f \sim g$ if $f \preceq g$ and $g \preceq f$. We refer to the $\sim$ equivalence class of f as the *growth* of f and denote it $[f]$. Then $\preceq$ defines a partial order on the growth classes of functions $\mathbb{N} \to \mathbb{R}^+$.

We make some definitions and easy observations about growth which will be used in the sequel without comment. All polynomials of degree d have the same growth, namely $[n^d]$ which we call *polynomial of degree d*. All exponential functions of the form a^n with $a > 1$ a real number have growth $[2^n]$ which we call *exponential*. Clearly, the conditions of growth being polynomial or exponential are mutually exclusive. Growth which is either polynomial or exponential is called *alternative* and growth which is neither polynomial nor exponential is called *intermediate*. Finally we have

Theorem 7.1.12. *Suppose that for some real numbers $a, h > 0$, $b, c \geq 0$ and for all sufficiently large $n \in N$ we have $g(n) = hf(an + b) + c$, then $[f] = [g]$.*

We now recall the notion of growth of a semigroup. Let S be a semigroup, A a set of generators for S and $\natural_A : A^+ \to S$ the natural homomorphism. For each $x \in S$ define the *length* $l(x)$ of x to be the least length of a word $w \in A^+$ such that $\natural_A(w) = x$. The *growth function* of S with respect to A is defined in [24] by

$$g_{S,A}(n) = |\{x \in S \mid l(x) \leq n\}|.$$

When S and A are understood, the growth function will be referred to simply as g. It is not difficult to see that the $\sim$-class of the growth function is independent of the generating set A so we can use *growth of the semigroup* to mean the $\sim$-class of any of its growth functions.

Finally we define the notion of growth for a formal language. Let $L \subseteq A^*$ be a language. The *growth function* h_L of L is given by defining $h_L(n)$ to be the number of words of L of length at most n. Then the *growth* of L is $[h_L]$.

7.1.5 Growth of a language of unique normal forms

One may also define the growth function of S with respect to A by

$$g(n) = |\natural_A(\{w \in A^+ \mid |w| \leq n\})|$$

and it is an easy exercise to see that this definition is equivalent to the previous one. Let L be a language of unique normal forms for S over A. Then $\natural_A$ is injective on the elements of L so that

$$\begin{aligned} h_L(n) &= |\natural_A(\{w \in L \mid |w| \leq n\})| \\ &\leq |\natural_A(\{w \in A^+ \mid |w| \leq n\})| \\ &= g(n). \end{aligned}$$

Therefore, noting that any semigroup has at least polynomial growth and at most exponential growth, we have

Theorem 7.1.13. *The growth of a language of unique normal forms for a semigroup S is bounded above by the growth of S. In particular, if S has polynomial growth, then any language of unique normal forms for S has polynomial growth, and if a language of unique normal forms for S has exponential growth, then so does S.*

Considering this theorem, a number of questions immediately spring to mind: When are the growth of the semigroup and the growth of its language of normal forms in the same class? The growth of the number of paths in a graph is known to be alternative [30], and therefore the growth of a regular language is alternative – is the growth of a rational semigroup necessarily alternative? In [30] it is shown that the growth of any algebra with finite Gröbner basis is alternative.

7.2 The monogenic free inverse semigroup is not rational

There appears to be consensus among workers in the area of automatic semigroups that it is more difficult to show that a semigroup is not automatic than to show that it is (which is usually a matter of exhibiting an automatic structure for it). In this section we use the fact that the growth of the free monogenic inverse semigroup is polynomial to show that it is not a rational semigroup (and therefore not automatic).

In [5, Chapter 8], it is shown that nilpotent groups are not automatic, and that proof also exploits the fact that nilpotent groups have polynomial growth. Nilpotent groups are, nevertheless, rational. As mentioned by Sims in [26], they have finite confluent rewriting systems under the basic wreath product ordering and it is a simple exercise in the theory of automata that this implies the existence of a regular set of unique normal forms.

7.2.1 Finite state machines

We start with some general facts about finite state machines, a construction used in the subsequent argument. A *finite state machine* consists of a finite set Δ of states, a finite set A of *input letters* and a function $\Gamma : \Delta \times A \to \Delta$ describing the *state transitions*. We extend Γ to a (right) monoid action of A^* on Δ. Denoting by ϵ the empty word in A^*, $\Gamma(\_, \epsilon)$ is therefore the identity on Δ. There is a distinguished state $i \in \Delta$ called the *initial* state and a subset $T \subseteq \Delta$ of *terminal* states. We will usually identify the state machine with its transition function. We also consider the *state graph* of the machine, which has vertex set Δ and an edge from s to t labelled by $a \in A$ if $\Gamma(s, a) = t$.

A word $w \in A^*$ is said to be *accepted* by Γ if $\Gamma(i, w) \in T$. A state $s \in \Delta$ is said to be *accessible* if there is some $w \in A^*$ such that $\Gamma(i, w) = s$ and *coaccessible* if there is a word $w \in A^*$ such that $\Gamma(s, w) \in T$.

The state graph of a state machine influences the growth of the language accepted by the machine in the following way.

Theorem 7.2.1. *Suppose the state graph of the state machine Γ has two distinct cycles on an accessible and coaccessible state. Then the language accepted by Γ has exponential growth.*

PROOF: Recall that a cycle in a graph on the vertex s is a path from s to itself passing through no other vertex twice.

Let s be the state with two distinct cycles in the state graph. Since s is accessible and coaccessible, there are words $u, v \in A^*$ such that $\Gamma(i, u) = s$ and $\Gamma(s, v) \in T$. Since the two cycles on s are distinct, there are distinct words $w_1, w_2 \in A^+$ (which are not prefixes of one another) which label the edges of the cycles, such that all words determined by the regular expression $u\{w_1, w_2\}^*v$ are accepted by Γ. Let $l = \mathrm{LCM}(|w_1|, |w_2|)$ and fix $p_1, p_2 \in \mathbb{N}$ such that $l = |w_1|p_1 = |w_2|p_2$. Then the number of words accepted by Γ of length $m = |u| + |v| + lk$ is at least 2^k. Namely, they contain the set of words given by the regular expression $u\{w_1^{p_1}, w_2^{p_2}\}^k v$, all of which are distinct.

Therefore, if the language accepted by the automaton has growth h, we have that $h(m) \geq 2^k = 2^{(m-|u|-|v|)/l}$ as required. ■

Theorem 7.2.1 is an automaton theoretic formulation of the fact that a language not being textitsimply starred (described by a regular expression in which the star operator is only applied to singletons) implies that it has exponential growth, a fact explained in [5, Section 1.3]. The next lemma dictates the form of words in a regular language with polynomial growth. In the terminology of [5] one would say that a regular language with polynomial growth is simply starred.

Lemma 7.2.2. *Let L be a regular language with polynomial growth accepted by some automaton Γ. L consists of precisely the words of the form*

$$u_1 v_1^{h_1} u_2 v_2^{h_2} \dots u_m v_m^{h_m} u_{m+1}$$

where $u_1 \dots u_{m+1}$ labels a cycle free path from the initial state of Γ to a terminal state, $h_i \geq 0$ for all i, $u_2, \dots, u_m$ are nonempty, and each v_i labels a cycle in the state graph on $\Gamma(i, u_1 \dots u_i)$.

PROOF: The result follows as a corollary of Theorem 7.2.1. Since L has polynomial growth, the state graph of Γ has no two cycles on a single accessible and coaccessible state. ■

7.2.2 The monogenic free inverse semigroup

For the remainder of Section 7.2 let FI_x denote the free monogenic inverse semigroup with (semigroup) generating set $\{x, x^{-1}\}$. We pause now to recall some simple facts and standard definitions about this semigroup. The reader requiring elucidation of the following development is referred to [18].

Let $\overline{}$ denote the homomorphism of FI_x onto the free group F_x of rank 1 defined by taking any word in $\{x, x^{-1}\}^+$ and freely reducing it, that is to say, cancelling xx^{-1} and $x^{-1}x$. For example, $\overline{xxx^{-1}x} = x^2$.

It is a consequence of the graph representation of free inverse semigroups (see [18, VIII.3]) that FI_x may be identified with the set of triples $(i, j, k) \in \mathbb{Z}^3$ such that $i < j$ and 0 and k are contained in the contiguous interval $[i, j]$. In particular, $i \leq 0 \leq j$. The product $(i, j, k) * (i', j', k')$ is then $(\min(i, k + i'), \max(j, k + j'), k + k')$. Let $\natural : \{x, x^{-1}\}^+ \to FI_x$ be the natural homomorphism mapping words to triples. This map is completely defined by setting $\natural(x) = (0, 1, 1)$ and $\natural(x^{-1}) = (-1, 0, -1)$.

It is a useful intuitive device to regard a triple as described above as a segment $[i, j]$ of $\mathbb{Z}$ with a distinguished element k. Then reading any word from left to right defines a path, starting at 0 and moving a step to the left every time x^{-1} is read, and a step to the right every time x is read. Then a word w such that $\natural(w) = (i, j, k)$ defines a path starting at 0, whose meanderings in the number line take it at most $|i|$ places left of zero and at most j places right of zero, finally ending at position k. Composing with another word v with $\natural(v) = (i', j', k')$ we start at k and meander at most $|i'|$ places to the left of k, j' places to the right of k and end up k' places to the right of k.

More formally, set $\text{lex}(w) = \min\{i \mid \overline{u} = x^i, u \text{ a prefix of } w\}$ and refer to it as the *left extremum* of w's path through $\mathbb{Z}$. Similarly define $\text{rex}(w) = \max\{i \mid \overline{u} = x^i, u \text{ a prefix of } w\}$ (the *right extremum*) and the *endpoint* given by $\overline{w} = x^{\text{end}(w)}$. With this notation we now have $\natural(w) = (\text{lex}(w), \text{rex}(w), \text{end}(w))$.

An immediate consequence of the discussion above is that

Theorem 7.2.3. *Let w be a word in $\{x, x^{-1}\}^+$. The following conditions are equivalent:*

- $\natural(w)$ *is idempotent;*
- $\overline{w} = 1$*;*
- $\natural(w) = (i, j, 0)$ *for some* $i, j \in Z$.

Finally we quote a well known result mentioned in [24] which is at the core of the proof of Theorem 7.2.7.

Theorem 7.2.4. *The free monogenic inverse semigroup has cubic growth.*

7.2.3 Proof of Theorem 7.2.7

For the remainder of this section we derive some lemmas under the assumption that FI_x is rational so that the proof proper is a proof by contradiction.

Suppose that L is a regular language of unique normal forms for FI_x over the alphabet $\{x, x^{-1}\}$, and let Γ be a finite state machine with n states accepting precisely the words of L.

Since FI_x has polynomial growth (by Theorem 7.2.4), L also has polynomial growth, so that each word of L may be written in the form described in Lemma 7.2.2. In particular, any word in L is of the form

$$u_1 v_1^{h_1} u_2 v_2^{h_2} \dots u_m v_m^{h_m} u_{m+1} \tag{7.1}$$

where,

- $u_1 \dots u_{m+1}$ describes a cycle free path in the state graph of Γ from the initial state to a terminal state;

- $u_2, \dots, u_m$ are nonempty;

- $m \leq n$;

- $\natural(v_i)$ is not idempotent, for otherwise the word obtained by increasing h_i by one, which is also accepted by Γ would represent the same element of FI_x contradicting uniqueness.

Let w be any word in L. Then w may be factored not only as in (7.1) but also as abc where $\mathrm{end}(a)$ and $\mathrm{end}(ab)$ are the opposite extrema of w's path. That is, either $\overline{a} = x^{\mathrm{lex}(w)}$ and $\overline{ab} = x^{\mathrm{rex}(w)}$, or $\overline{a} = x^{\mathrm{rex}(w)}$ and $\overline{ab} = x^{\mathrm{lex}(w)}$.

However it may happen (inconveniently for our purposes) that a or b ends within one of the v_i. The next lemma shows that we may choose a, b and c so that their boundaries are out of the v_i but where $\mathrm{end}(a)$ and $\mathrm{end}(ab)$ are still 'not too far' from the extrema of w's path.

Lemma 7.2.5. *Let $w \in L$. Then w may be factored as abc and also as in (7.1) so that*

- $a = u_1 {v_1}^{h_1} \dots u_{j-1} {v_{j-1}}^{h_{j-1}} {u_j}'$;
- $b = {u_j}'' {v_{j+1}}^{h_{j+1}} \dots u_{k-1} {v_{k-1}}^{h_{k-1}} {u_k}'$;
- $c = {u_k}'' {v_{k+1}}^{h_{k+1}} \dots u_m {v_m}^{h_m} u_{m+1}$;

and so that $\mathrm{end}(a)$ is within n of the lower extremum of w's path in $\mathbb{Z}$ and $\mathrm{end}(ab)$ is within n of the upper extremum, or vice versa.

PROOF: We prove the lemma for the case that w may be factored as $a'b'c'$ with $\mathrm{end}(a')$ the lower extremum and $\mathrm{end}(a'b')$ the upper extremum. The other case is similar.

If a' ends within u_j for some j then put $a = a'$. Otherwise, $a' = u_1 {v_1}^{h_1} \dots u_j {v_j}^{h} v_j'$ for some prefix v_j' of v_j.

Now if $\overline{v_j}$ is a negative power of x, then h must be $h_j - 1$, in which case put $a = u_1 {v_1}^{h_1} \dots u_j {v_j}^{h_j}$. Then $\overline{a}$ cannot be more than an $(n-1)$th power of x

greater than $\overline{a'}$ since no state appears more than once going from $\Gamma(i, a')$ to $\Gamma(i, a)$ since it traces the last part of a cycle in the state graph of Γ.

If, on the other hand, $\overline{v_j}$ is a positive power of x, then $h = 0$ so we can let $a = u_1 {v_1}^{h_1} ... u_{j-1}$. Again $\overline{a}$ can differ from $\overline{a'}$ by no more than an $(n-1)$th power of x.

Now we have $w = ab''c'$ where $ab'' = a'b'$, defines a path in $\mathbb{Z}$ with endpoint the right extremum of w's path. We now have $\text{lex(w)} \leq \text{end}(a) < \text{lex(w)} + n$, as required. Of couse, we still have $\text{end}(ab'') = \text{rex}(w)$.

If ab'' ends within u_k, put $b = b''$ and $c = c'$ and we are done. Otherwise, b'' is the word starting at the end of a and ending with $u''_k v^h_k v'_k$ for some prefix v'_k of v_k and u''_k is some (possibly empty) suffix of u_k.

If $\overline{v_k}$ is a negative power of x then, $h = 0$. Truncate b'' at the end of u''_k to produce b. If $\overline{v_k}$ is a positive power of x then h is $h_k - 1$. Append the rest of v_k to form b

In either case, noting that $\text{end}(b'') - n < \text{end}(b)$, we still have $\text{rex}(w) - n < \text{end}(ab) \leq \text{rex}(w)$. ■

It is now shown that if $w \in L$ represents a 'large enough' element of FI_x, then as Γ accepts w, each of the factors a, b and c determined by Lemma 7.2.5 traverses a cycle in the state graph of Γ. The astute reader will recognize this as a thinly disguised Pumping Lemma [8].

Lemma 7.2.6 (Pumping Lemma). *Let w be an element of L with $\natural(w) = (p, q, 0)$. If $p < -2n$ and $q > 2n$ then w factors as in (7.1), and for some $i_1 < i_2 < i_3$, the factors $\overline{v_{i_1}}$, $\overline{v_{i_2}}$ and $\overline{v_{i_3}}$, are nonzero powers of x which alternate in sign.*

PROOF: We can write $w = abc$ as in the statement of Lemma 7.2.5 with $\text{end}(a)$ within n of the lower extremum of w's path and $\text{end}(ab)$ within n of the upper extremum, or vice versa. Without loss of generality we assume the former.

To begin with, consider $a = u_1 {v_1}^{h_1} \ldots u_{j-1} {v_{j-1}}^{h_{j-1}} {u_j}'$. Now $u_1 u_2 \ldots u_{j-1} u'_j$ traces out a path in the state graph of Γ which does not visit the same state twice hence $\overline{u_1 \ldots u'_j}$ is a power of x which is between $-n$ and n. But $\overline{a}$ is a power of x^{-1} which is greater than n. Thus there is some $1 \leq i_1 < j$ with $\overline{v_{i_1}}$ a negative power of x and $h_{i_1} > 0$.

Similarly, $\overline{b}$ is a power of x which is greater than $2n$, which implies that there is some $j \leq i_2 < k$ with $\overline{v_{i_2}}$ a positive power of x and $h_{i_2} > 0$.

An identical argument assures us that there is some $k \leq i_3 \leq m$ with $\overline{v_{i_3}}$ a negative power of x and $h_{i_3} > 0$. ■

Finally we are in a position to prove main theorem of this section.

Theorem 7.2.7. *The monogenic free inverse semigroup is not rational.*

PROOF: Suppose by way of contradiction that FI_x is rational. Then by Lemma 7.1.3 it must have a regular language of unique normal forms over the generating set $\{x, x^{-1}\}$. Let L be such a supposed language and Γ a finite state machine which accepts precisely the words of L. Let n be the number of states of Γ. Since L is neither $\{x, x^{-1}\}^*$ nor $\emptyset$, n must be at least 2.

Under these assumptions we proceed to exhibit two words in L with the same image under $\natural$ contradicting uniqueness.

Let w be the unique element of L with $\natural(w) = (-2n-1, 2n+1, 0)$. Then w satisfies the conditions of Lemma 7.2.6. So without loss of generality we may write $w = u_1 v_1^{h_1} u_2 v_2^{h_2} \ldots u_m v_m^{h_m} u_{m+1}$ as in (7.1) and assume that there are $i_1 < i_2 < i_3$ with:

- $\overline{v_{i_1}} = x^{f_1}$ and $f_1 < 0$;
- $\overline{v_{i_2}} = x^{f_2}$ and $f_2 > 0$;
- $\overline{v_{i_3}} = x^{f_3}$ and $f_3 < 0$; and

h_{i_1}, h_{i_2} and h_{i_3} nonzero. Let $\delta_2, \delta_3 > 0$ be the unique integers such that

$$f_2\delta_2 = -f_3\delta_3 = \operatorname{lcm}(f_2, -f_3). \tag{7.2}$$

Observe that $0 < \delta_2 \leq -f_3 \leq |v_{i_3}| \leq n$ and that similarly $0 < \delta_3 \leq n$. Let $\lambda = \operatorname{lcm}(-f_1, f_2, -f_3)$ (a positive integer). Then set

$$\begin{aligned} \alpha &= \frac{8n^2\lambda}{-f_1} \\ \beta &= \frac{4n^2\lambda}{f_2} \\ \gamma &= \frac{2n^2\lambda}{-f_3}. \end{aligned}$$

By the fact that $n \geq 2$, $\beta > n \geq \delta_2$ and $\gamma > b \geq \delta_3$. Define

$$\begin{aligned} w_1 &= u_1 u_2 \dots u_{i_1} v_{i_1}^{\alpha} u_{i_1+1} \dots u_{i_2} v_{i_2}^{\beta} u_{i_2+1} \dots u_{i_3} v_{i_3}^{\gamma} u_{i_3+1} \dots u_{m+1} \\ w_2 &= u_1 u_2 \dots u_{i_1} v_{i_1}^{\alpha} u_{i_1+1} \dots u_{i_2} v_{i_2}^{\beta-\delta_2} u_{i_2+1} \dots u_{i_3} v_{i_3}^{\gamma-\delta_3} u_{i_3+1} \dots u_{m+1}. \end{aligned}$$

The construction by which we arrive at the factorization (7.1) ensures that w_1 and w_2 are both accepted by Γ and are therefore in L. It only remains to show that $\natural(w_1) = \natural(w_2)$. The equality holds if the endpoints are equal (which is equivalent to showing that $\overline{w_1} = \overline{w_2}$) and that the left and right extrema are equal.

Now by commutativity of F_x,

$$\begin{aligned} \overline{w_1} &= \overline{w_2 v_{i_2}^{\delta_2} v_{i_3}^{\delta_3}} \\ &= \overline{w_2} x^{f_2 \delta_2} x^{f_3 \delta_3} \\ &\quad \text{but by (7.2)} \\ &= \overline{w_2} x^{-f_3 \delta_3} x^{f_3 \delta_3} \\ &= \overline{w_2} \end{aligned}$$

as required. Now we calculate the left and right extrema of the paths of w_1 and w_2 in $\mathbb{Z}$. A helpful observation for the following calculations is that if $\overline{v}$ is a positive power of x, then for all $k > 0$, $\text{lex}(v^k) = \text{lex}(v)$ and similarly, if $\overline{v}$ is a negative power of x, then for all $k > 0$, $\text{rex}(v^k) = \text{rex}(v)$. Note also that

$$\text{lex}(uv) = \min(\text{lex}(u), \text{end}(u) + \text{lex}(v)) \tag{7.3}$$

and

$$\text{rex}(uv) = \max(\text{rex}(u), \text{end}(u) + \text{rex}(v)). \tag{7.4}$$

Let a_1 be the prefix of w_1 given by $u_1 u_2 \dots u_{i_1} v_{i_1}^{\alpha} u_{i_1+1} \dots u_{i_2} v_{i_2}^{\beta}$ and let a_2 be the prefix of w_2 given by $u_1 u_2 \dots u_{i_1} v_{i_1}^{\alpha} u_{i_1+1} \dots u_{i_2} v_{i_2}^{\beta-\delta_2}$. Choose b_1 and b_2 so that $w_1 = a_1 b_1$ and $w_2 = a_2 b_2$. Since $\overline{v_{i_2}}$ is a positive power of x, we can easily deduce that

$$\begin{aligned} \text{lex}(a_1) &= \text{lex}(a_2) \\ &= \text{lex}(u_1 u_2 \dots u_{i_1} v_{i_1}^{\alpha} u_{i_1+1} \dots u_{i_2} v_{i_2}) \\ &< n - 8n^2\lambda. \end{aligned}$$

To determine lower bounds on $\text{end}(a_1), \text{end}(a_2), \text{lex}(b_1)$ and $\text{lex}(b_2)$, we assert only that $\text{end}(u_1 \ldots u_{i_2}) > -n$ and $\text{end}(u_{i_2+1} \ldots u_{m+1}) > -n$. Thus,

$$\begin{aligned}
\text{end}(a_1) &> -n + \alpha f_1 + \beta f_2 \\
&= -n - 8n^2\lambda + 4n^2\lambda \\
&= -4n^2\lambda - n, \text{ and similarly,} \\
\text{end}(a_2) &> -4n^2\lambda - n - \delta_2 f_2; \\
\text{lex}(b_1) &> -2n^2\lambda - n; \text{ and} \\
\text{lex}(b_2) &> -2n^2\lambda - n - \delta_3 f_3; \\
&= -2n^2\lambda - n + \delta_2 f_2.
\end{aligned}$$

We show that $\text{lex}(a_1) < \text{end}(a_1) + \text{lex}(b_1)$ and $\text{lex}(a_2) < \text{end}(a_2) + \text{lex}(b_2)$ which proves (by (7.3)) that $\text{lex}(w_1) = \text{lex}(a_1 b_1) = \text{lex}(a_1) = \text{lex}(a_2) = \text{lex}(a_2 b_2) = \text{lex}(w_2)$ as required. Now it a simple matter of arithmetic to show that if either of these two inequalities didn't hold, then we would have $2n^2\lambda - 3n \leq 0$. But this is only true for values of n between 0 and $\frac{3}{2\lambda}$. Since $\lambda \geq 1$ we have shown a contradiction since our automaton must have at least 2 states. From this we conclude that the left extrema of w_1 and w_2 are the same.

To complete the proof of the theorem, it is now shown in a similar way that the right extrema of w_1 and w_2 are the same. Let $a = u_1 u_2 \ldots u_{i_1} v_{i_1}^{\alpha}$ and once again choose b_1 and b_2 so that $w_1 = ab_1$ and $w_2 = ab_2$. We claim that $\text{rex}(w_1) = \text{rex}(w_2) = \text{rex}(a)$.

A priori, $\text{rex}(a) \geq 0$. In the same manner as the previous part of the proof, we calculate:

$$\begin{aligned}
\text{end}(a) &< -8n^2\lambda + n; \text{ and} \\
\text{rex}(b_1), \text{rex}(b_2) &< 4n^2\lambda + n.
\end{aligned}$$

If $\text{rex}(w_1)$ or $\text{rex}(w_2)$ are not equal to $\text{rex}(a)$ then (7.4) implies that $\text{rex}(a) < \text{end}(a) + \text{rex}(b_1)$ or $\text{rex}(a) < \text{end}(a) + \text{rex}(b_2)$. In either case we would have $-4n^2\lambda + 2n \geq 0$, which only occurs for values of n between 0 and $\frac{1}{2\lambda}$, once again contradicting the fact that the automaton has at least 2 states. Thus the right extrema of w_1 and w_2 are the same.

This completes the proof that no regular language of normal forms for FI_x can have uniqueness. ∎

7.3 Application and Discussion

The remarks in Section 1 together with the theorem of Section 2 allow us to draw some useful conclusions and conjecture further results.

In contrast with finitely generated free groups and free semigroups which are both easily seen to be automatic and therefore rational

Theorem 7.3.1. *No free inverse semigroup is rational. Therefore no free inverse semigroup is automatic.*

PROOF: Let FI_X denote the free inverse semigroup on a finite set X and let $x \in X$. Then define a map $\phi : X \to \{x, 0\}$ by

$$\phi(y) = \begin{cases} x & \text{if } y = x \\ 0 & \text{otherwise} \end{cases}$$

and extend it to a Rees quotient map $\phi : FI_X \to FI_x^0$. If FI_X were rational then Theorem 7.1.10 would imply that FI_x^0, and by Theorem 7.1.9, that FI_x was rational – a contradiction. ■

Together with Theorem 7.1.11 this shows that

Corollary 7.3.2. *No semigroup can be rational (nor, therefore, automatic) if it is a free product of a free inverse semigroup with another semigroup.*

The class of semigroups which we now know not to be rational is not contained within the class of semigroups with polynomial growth, since the free inverse semigroup on more than one generator has exponential growth. This fact is somewhat intriguing since the proof of Theorem 7.2.7 is so dependent on the growth of FI_x.

An obvious question which arises is whether a free inverse semigroup may embed in any rational semigroup, for if not, FI_X would be an interesting semigroup satisfying the third condition in the definition of a Markov property, while still having solvable word problem.

Another class of inverse semigroups closely entwined with the present thread of discourse are defined in [13]:

Theorem 7.3.3. *Suppose S is a finitely presented Rees quotient of a free inverse semigroup with polynomial growth. Then the following conditions are equivalent:*

- *S is infinite;*
- *S contains a free monogenic inverse subsemigroup;*
- *S has growth of degree at least* 3.

We conjecture that the semigroups defined by Theorem 7.3.3 are not rational.

As a final remark, the observations of Section 7.1.2 recall a lecture given by Professor Rick Thomas at the conference CGAMA at Heriot-Watt University, Edinburgh in July 1998 [28]. For a finitely presented group G the set $W(G)$ of words representing the identity of G was considered. A number of theorems relating the position of $W(G)$ in the formal language hierarchy with the algebraic structure of G were cited. We consider it a promising line of inquiry to investigate the algebraic properties of groups and semigroups which are known to have a language of unique normal forms in the various strata of the language hierarchy.

Bibliography

1. S. I. Adjan, On algorithmic problems in effectively complete classes of groups (Russian), *Dokl. Akad. SSSR.* **123** (1958), 13-16.

2. S. Burris and H. P. Sankappanavar, *A Course in Universal Algebra*, Springer-Verlag 1981.

3. C. M. Campbell, E. F. Robertson, N. Ruškuc, R. M. Thomas, On semigroups defined by Coxeter-type presentations, *Proc. Royal Edin. Soc. Sect. A* **125** (1995) no. 5, 1063-1075.

4. C. M. Campbell, E. F. Robertson, N. Ruškuc, R. M. Thomas, Automatic Semigroups, *Theoret. Comput. Sci.* (to appear).

5. J. W. Cannon, D. B. A. Epstein, D. F. Holt, S. V. F. Levy, M. S. Paterson, W. P. Thurston, *Word processing in groups* (Jones and Barlett Publishers, Boston, Massachusetts, 1992).

6. A. J. Duncan, E. F. Robertson, N. Ruškuc, Automatic monoids and change of generators, *Math. Proc. Cambridge Philos. Soc.* **127** (1999), 403-409.

7. R. Gilman, Groups with a rational cross-section, in S. M. Gersten and J. R. Stallings (eds.), *Combinatorial Group Theory and Topology*, (Princeton University Press, Princeton, New Jersey, 1987), 175-183.

8. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, (Addison-Wesley Publishing Co., Reading, Mass., 1979).

9. J. M. Howie, *Fundamentals of Semigroup Theory*, Clarendon Press, Oxford, 1995.

10. J. M. Howie, *Automata and Languages*, Oxford Science Publications, (1991).

11. D. L. Johnson, *Topics in the Theory of Group Presentations*, LMS Lecture Notes 42, Cambridge University Press (1980).

12. A. Jura, Coset enumeration in a finitely presented semigroup, *Canad. Math. Bull.* **21** (1978), 37-46.

13. J. Lau, Degree of Growth of some Inverse Semigroups, *J. Algebra* **204** (1998), no.2, 426-439.

14. A. Markov, On the impossibility of certain algorithms in the theory of associative systems, *Dokl. Accad. Sci. SSSR* **55** (1947), 583-586.

15. A. Markov, On the impossibility of algorithms for recognising some properties of associative systems, *Dokl. Accad. Sci. SSSR* **77** (1951), 953-956.

16. W. D. Munn, Free inverse semigroups, *Proc. London Math. Soc,* (3) **29** (1974) 385-404.

17. B. H. Neumann, Some remarks on semigroup presentations, *Canad. J. Math.* **19** (1967), 1018-1026.

18. M. Petrich, *Inverse Semigroups*, Wiley Interscience Series (1984).

19. M. O. Rabin, Recursive unsolvability of group theoretic problems, *Ann. Math.* **67** (1958), 172-194.

20. C. Reutenauer, Une topologie du monoide libre, *Semigroup Forum* **18** (1979), 33-49.

21. E. F. Robertson and R. M. Thomas, On a class of semigroups with symmetric presentations, *Semigroup Forum* **46** (1993) 286-306.

22. E. F. Robertson and Y. Ünlü, On semigroup presentations, *Proc. Edinburgh Math. Soc.* **36** (1992), 55-68.

23. N. Ruškuc, On finite presentability of monoids and their Schtzenberger groups, *Pacific J. Math* **195** (2000) no. 2, 487–509.

24. L. M. Schneerson and D. Easdown, Growth and existence of identities in a class of finitely presented inverse semigroups with zero, *Internat. J. Algebra Comput.*, **6**(1) (1996), 105-121.

25. M. Schönert *et al.* (1995). GAP - *Groups, Algorithms and Programming*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, fifth edition. See http://www-gap.dcs.st-and.ac.uk/ gap.

26. C. C. Sims, *Computation with finitley presented groups*, Encyclopedia of Mathematics and its Applications, **48** (Cambridge University Press, Cambridge, 1994).

27. J. B. Stephen, Presentations of inverse monoids, *J. Pure Appl. Algebra* **63** (1990) 81-112.

28. R. Thomas, Groups and formal languages, Lecture at CGAMA, Heriot-Watt, July 1998.

29. J. A. Todd and H. S. M. Coxeter, A practical method for enumerating cosets of a finite abstract group, *Proc. Edinburgh Math. Soc.* **5** (1936), 26-34.

30. V. A. Ufnarovskij, *Combinatorial and Asymptotic Methods in Algebra*, Encyclopedia of Mathematical Sciences, **57**, (Springer, Berlin, 1995).

31. V. V. Wagner, Generalised Groups, *Doklady Akad. Nauk* SSSR **84** (1952), 1119-1122.

32. A. Yamamura, HNN Extensions of Inverse Semigroups and Applications, *Internat. J. Algebra Comput.* **7**, No. 5 (1997) 605-624.

Appendix

```
###########################################################################
##
#F  RClassCosetTable( <M>, <word>, <table>, <coset> )
##                                                          . . coset table of
##                                                          R-class generated by
##                                                          word.
RClassCosetTable:= function(M, word, table, identity)

 local a, i, c, k, r, d, n, FLAG,
       active,
       gens, rels,
       leng,
       tidytable,
       stack,
       replace,
       complete,
       parsymrep,
       numberUndel,
       list,
       eqnTrace,
       ideNtify,
       newCoset;

  newCoset:=function(c,a)

##modifies table and returns the new coset

    Add(table,List(leng,x->0));
    table[c][Position(gens,a)]:=Length(table);
    table[Length(table)][((Position(gens,a)+n/2-1) mod n)+1]:=c;
    Add(replace,0);
    Add(complete,false);
    active:=active+1;

    return Length(table);

  end;

  eqnTrace:=function(c,r)

#trace the relation r starting at coset c

    local s,t,u,a,b,lflag,rflag;

#trace through lefthand side
    s:=c;
    a:=1;
    while not s=0 and a<=LengthWord(r[1]) do
      u:=s;
      s:=table[s][Position(gens,Subword(r[1],a,a))];
      a:=a+1;
    od;
    if s=0 then
      lflag:=false;
      s:=u;
      a:=a-1;
    else
      lflag:=true;
    fi;

#trace through righthand side
    t:=c;
    b:=1;
    while not t=0 and b<=LengthWord(r[2]) do
      u:=t;
      t:=table[t][Position(gens,Subword(r[2],b,b))];
```

```
      b:=b+1;
    od;
    if t=0 then
      rflag:=false;
      t:=u;
      b:=b-1;
    else
      rflag:=true;
    fi;

#check relation
    if not lflag and rflag then
#trace through left of <r>
      while a<=LengthWord(r[1]) do
        s:=newCoset(s,Subword(r[1],a,a));
        a:=a+1;
      od;
      Add(stack,[Minimum(s,t),Maximum(s,t)]);
    elif lflag and not rflag then
#trace through right of <r>
      while b<=LengthWord(r[2]) do
        t:=newCoset(t,Subword(r[2],b,b));
        b:=b+1;
      od;
      Add(stack,[Minimum(s,t),Maximum(s,t)]);
    elif lflag and rflag and not s=t then
      Add(stack,[Minimum(s,t),Maximum(s,t)]);
    fi;

    ideNtify();

  end;

  ideNtify:=function()

#coset collapse

    local s,t,a,i,u,v;

    while not stack=[] do
      FLAG:=true;
      s:=stack[Length(stack)];
      Unbind(stack[Length(stack)]);
      while replace[s[1]]>0 do
        s[1]:=replace[s[1]];
      od;
      while replace[s[2]]>0 do
        s[2]:=replace[s[2]];
      od;
# do the identification.
      if not s[1]=s[2] then
        for i in [1..Length(table)] do
          if replace[i]=0 and not i=s[2] then
            for a in leng do
              if table[i][a]=s[2] then
                table[i][a]:=s[1];
                v:=table[s[1]][((a+n/2-1) mod n)+1];
                if v=0 then
                  table[s[1]][((a+n/2-1) mod n)+1]:=i;
                else
                  Add(stack,[Minimum(i,v),Maximum(i,v)]);
                fi;
              fi;
            od;
          fi;
        od;
```

```
#modify table
        for a in leng do
          v:=table[s[2]][a];
          if v>0 then
            u:=table[s[1]][a];
            if u=0 then
              table[s[1]][a]:=v;
              table[v][((a+n/2-1) mod n)+1]:=s[1];
            else
              Add(stack,[Minimum(u,v),Maximum(u,v)]);
            fi;
          fi;
        od;
#modify stack
        for t in stack do
          if t[1]=s[2] then
            t[1]:=s[1];
          elif t[2]=s[2] then
            t[2]:=s[1];
          fi;
        od;

        active:=active-1;
        replace[s[2]]:=s[1];

        if s[2]=identity then
          identity:=s[1];
        fi;

      fi;
    od;

  end;

  #initialize
  gens:=Copy(M.generators);
  Append(gens,M.inverses);
  n:=Length(gens);
  leng:=[1..Length(gens)];
  rels:=M.relations;
  stack:=[];
  replace:=List([1..Length(table)],x->0);
  complete:=List([1..Length(table)],x->false);
  FLAG:=true;
  active:=Length(table);

#main routine
  while FLAG do
    FLAG:=false;
    c:=1;
    repeat
      for r in rels do
        if replace[c]=0 then
          eqnTrace(c,r);
        fi;
      od;
      complete[c]:=true;
      repeat
        c:=Position(complete,false);
        if c=false then c:=1;fi;
        if replace[c]>0 then
          complete[c]:=true;
        fi;
      until replace[c]=0 or not false in complete;
    until not false in complete;
    complete:=List([1..Length(table)],x->replace[x]>0);
```

```
  od;

#tidy up
  numberUndel:=[];
  k:=0;
  for i in [1..Length(table)] do
    if replace[i]>0 then
      k:=k+1;
    fi;
    Add(numberUndel,i-k);
  od;

  if not Set(replace)=[0] then
    tidytable:=[];
    for i in [1..Length(table)] do
      if replace[i]=0 then
        Add(tidytable,[]);
        for a in leng do
          if table[i][a]=0 then
            tidytable[Length(tidytable)][a]:=0;
          else
            tidytable[Length(tidytable)][a]:=numberUndel[table[i][a]];
          fi;
        od;
      fi;
    od;
  else
    tidytable:=table;
  fi;

  return rec(
             active:=active,
             representative:=word,
             identity:=numberUndel[identity],
#Number(replace,x->x=0),###Alternative return
#representative:=parsymrep,###Alternative return
             table:=tidytable);

end;
################################################################################
##
#F  Trace( <generators>,<table>,<word>,<coset> ) . . . word traces  to...?
##
Trace:=function( gens, table, word, coset )

  local a,c,l;

  l:=Length(gens);

  c:=coset;
  for a in List(word) do
    c:=table[c][Position(gens,a)];
    if c=0 then
      return 0;
    fi;
  od;

  return c;

end;

################################################################################
##
#F  Enumerate( <M> ) . . . . . . . enumerates inverse monoid given by
##                                     the presentation M
Enumerate:=function(M)
```

```
  local RClasses,r,s,newtab,i,a,c,gens,l,flag;

  gens:=ShallowCopy(M.generators);
  Append(gens,M.inverses);
  l:=Length(gens);
  RClasses:=[];
  Add(RClasses,RClassCosetTable(M,IdWord,[List([1..l],x->0)],1));

#orbit algorithm
  for r in RClasses do
    for a in gens do
      c:=r.table[r.identity][((Position(gens,a)+l/2-1) mod l)+1];
      if c=0 then
        newtab:=Copy(r.table);
        Add(newtab,List([1..l],x->0));
        newtab[r.identity][((Position(gens,a)+l/2-1) mod l)+1]:=r.active+1;
        newtab[r.active+1][Position(gens,a)]:=r.identity;
        s:=RClassCosetTable(M,a*r.representative,newtab,r.active+1);
      else
        s:=rec(active:=r.active,representative:=a*r.representative,identity:=c,
               table:=Copy(r.table));
      fi;
      flag:=false;
      i:=1;
      while not flag and i<Length(RClasses)+1 do
        flag:=Trace(gens,RClasses[i].table,s.representative,
                    RClasses[i].identity)>0 and
              Trace(gens,s.table,RClasses[i].representative,s.identity)>0;
        i:=i+1;
      od;
      if not flag then
        Add(RClasses,s);
      fi;
    od;
  od;

  return rec(Size:=Sum(RClasses,x->x.active),
             NumberOfRClasses:=Length(RClasses),
             RClassTables:=List(RClasses,x->rec(Size:=x.active,
                                               Representative:=x.representative,
                                               Identity:=x.identity,
                                               Table:=x.table)));

end;


#############################################################################
##
#F  RClassAlg( <M>, <cong>, <word>, <type> ) . . .           returns coset table of
##                                                      R-class generated by
##                                                      word factored out by cong.
##                                                      Original RClass
##                                                      enumerater v. similar to
##                                                      main one. For support.
RClassAlg:= function(M, cong, word, type)

##type=0 - normal; type=1 - R/H;

 local a, c, i, k, r, d, n,
       active,
       gens, rels,
       leng,
       table,
       tidytable,
       stack,
```

```
      replace,
      complete,
      parsymrep,
      numberUndel,
      list,
      representative,
      eqnTrace,
      ideNtify,
      newCoset,
      Inverted,
      Cancelled,
      trace,
      AddNewCong;

  Inverted:=function(w)

##returns the inverse of the word w

    local nw,a;

    nw:=IdWord;
    for a in Reversed(List(w)) do
      nw:=nw*gens[((Position(gens,a)+n/2-1) mod n) +1];
    od;

    return nw;

  end;

  Cancelled:=function(w)

##given a word w this function cancels it as if it were a word in a group

    local nw,a;

    nw:=IdWord;
    for a in List(w) do
      if Position(gens,a)<=n/2 then
        nw:=nw*a;
      else
        nw:=nw*gens[Position(gens,a)-n/2]^-1;
      fi;
    od;

    return nw;

  end;

  newCoset:=function(c,a)

##modifies table and returns the new coset

    Add(table,List(leng,x->0));
    table[c][Position(gens,a)]:=Length(table);
    table[Length(table)][((Position(gens,a)+n/2-1) mod n)+1]:=c;
    Add(replace,0);
    Add(representative,representative[c]*a);
    Add(complete,false);
    active:=active+1;

    return Length(table);

  end;

  eqnTrace:=function(c,r)
```

```
    local s,t,u,a,b,lflag,rflag;

# does <c> trace through left of <r>?
    s:=c;
    a:=1;
    while not s=0 and a<=LengthWord(r[1]) do
      u:=s;
      s:=table[s][Position(gens,Subword(r[1],a,a))];
      a:=a+1;
    od;
    if s=0 then
      lflag:=false;
      s:=u;
      a:=a-1;
    else
      lflag:=true;
    fi;

# does <c> trace through right of <r>?
    t:=c;
    b:=1;
    while not t=0 and b<=LengthWord(r[2]) do
      u:=t;
      t:=table[t][Position(gens,Subword(r[2],b,b))];
      b:=b+1;
    od;
    if t=0 then
      rflag:=false;
      t:=u;
      b:=b-1;
    else
      rflag:=true;
    fi;

    if not lflag and rflag then
#trace through left of <r>
      while a<=LengthWord(r[1]) do
        s:=newCoset(s,Subword(r[1],a,a));
        a:=a+1;
      od;
    Add(stack,[Minimum(s,t),Maximum(s,t)]);
    elif lflag and not rflag then
#trace through right of <r>
      while b<=LengthWord(r[2]) do
        t:=newCoset(t,Subword(r[2],b,b));
        b:=b+1;
      od;
    Add(stack,[Minimum(s,t),Maximum(s,t)]);
    elif lflag and rflag and not s=t then
      Add(stack,[Minimum(s,t),Maximum(s,t)]);
    fi;

    ideNtify();

  end;


  ideNtify:=function()

#coset collapse

    local s,t,a,i,u,v;

    while not stack=[] do
      s:=stack[Length(stack)];
      Unbind(stack[Length(stack)]);
```

```
      while replace[s[1]]>0 do
        s[1]:=replace[s[1]];
      od;
      while replace[s[2]]>0 do
        s[2]:=replace[s[2]];
      od;
 # do the identification.
      if not s[1]=s[2] then
        for i in [1..Length(table)] do
          if replace[i]=0 and not i=s[2] then
            for a in leng do
              if table[i][a]=s[2] then
                table[i][a]:=s[1];
                v:=table[s[1]][((a+n/2-1) mod n)+1];
                if v=0 then
                  table[s[1]][((a+n/2-1) mod n)+1]:=i;
                else
                  Add(stack,[Minimum(i,v),Maximum(i,v)]);
                fi;
              fi;
            od;
          fi;
        od;

        for a in leng do
          v:=table[s[2]][a];
          if v>0 then
            u:=table[s[1]][a];
            if u=0 then
              table[s[1]][a]:=v;
              table[v][((a+n/2-1) mod n)+1]:=s[1];
            else
              Add(stack,[Minimum(u,v),Maximum(u,v)]);
            fi;
          fi;
        od;

        for t in stack do
          if t[1]=s[2] then
            t[1]:=s[1];
          elif t[2]=s[2] then
            t[2]:=s[1];
          fi;
        od;

        active:=active-1;
        replace[s[2]]:=s[1];

      fi;
    od;

  end;

  trace:=function(w)

#trace the word w starting coset 1

    local a,c;

    c:=1;
    for a in List(w) do
      c:=table[c][Position(gens,a)];
      if c=0 then
        return 0;
      fi;
    od;
```

```
    return c;

  end;

  AddNewCong:=function()

#R/H subroutine which adds new right congruence genertors

    local w,flag;

    flag:=false;

    for w in [2..Length(table)] do
      if replace[w]=0 and trace(representative[w]*word)>0
         and trace(Inverted(representative[w])*representative[w])>0 then
        Add(cong,[representative[w],IdWord]);
      fi;
    od;

    if flag then complete:=Copy(List(table,x->false));fi;

  end;

  #initialize
  gens:=Copy(M.generators);
  Append(gens,M.inverses);
  n:=Length(gens);
  leng:=[1..Length(gens)];
  rels:=M.relations;
  table:=[List(leng, x->0)];
  stack:=[];
  replace:=[0];
  representative:=[IdWord];
  complete:=[false];
  active:=1;

  if not word=IdWord then
    c:=1;
    for a in List(word) do
      if table[c][Position(gens,a)]=0 then
        c:=newCoset(c,a);
      else
        c:=table[c][Position(gens,a)];
      fi;
    od;
  fi;

  if type=1 then AddNewCong();fi;

#main routine
  c:=1;
  repeat
    for r in cong do
      eqnTrace(1,r);
    od;
    for d in [1..c] do
      if replace[d]=0 then
        for r in rels do
          eqnTrace(d,r);
        od;
      fi;
    od;
    complete[c]:=true;
    if type=1 then AddNewCong();fi;
    repeat
```

```
      c:=Position(complete,false);
      if c=false then c:=1;fi;
      if not replace[c]=0 then
        complete[c]:=true;
      fi;
    until replace[c]=0 or not false in complete;
  until not false in complete;

#tidy up
  numberUndel:=[];
  k:=0;
  for i in [1..Length(table)] do
    if replace[i]>0 then
      k:=k+1;
    fi;
    Add(numberUndel,i-k);
  od;

  if not Set(replace)=[0] then
    tidytable:=[];
    for i in [1..Length(table)] do
      if replace[i]=0 then
        Add(tidytable,[]);
        for a in leng do
          if table[i][a]=0 then
            tidytable[Length(tidytable)][a]:=0;
          else
            tidytable[Length(tidytable)][a]:=numberUndel[table[i][a]];
          fi;
        od;
      fi;
    od;
  else
    tidytable:=table;
  fi;

  if type=0 then
    return rec(Size:=active,
               Representative:=word,
#representative:=parsymrep,
               Table:=tidytable);
  else
    return rec(Size:=active,
               Representative:=word,
               RightCongruenceGenerators:=List(cong,x->x[1]),
#representative:=parsymrep,
               Table:=tidytable);
  fi;

end;

#############################################################################
##
#F      Inverted( <generators>, <word> ).  . Inverts word
##

Inverted:=function(gens,w)

  local n,nw,a;

  n:=Length(gens);
  nw:=IdWord;
  for a in Reversed(List(w)) do
    nw:=nw*gens[((Position(gens,a)+n/2-1) mod n) +1];
  od;
```

```
  return nw;

end;

################################################################################
##
#F      RClassEnumerate(<M>, <cong>, <word>)  .   .  . Standard Enumeration for
##                                                  R-class generated by word
##                                                  with the right congruence cong
##                                                  factored out
RClassEnumerate:=function(M, cong, word)

  cong:=List(cong,x->[x,IdWord]);

  return RClassAlg(M, cong, word, 0);

end;


################################################################################
##
#F      ROverHEnumerate(<M>, <cong>, <word>) .  .   . Enumerates the R-class generated
##                                                  by word the right congruence
##                                                  generated by H and cong
##                                                  factored out.
ROverHEnumerate:=function(M, cong, word)

  cong:=List(cong,x->[x,IdWord]);

  return RClassAlg(M, cong, word, 1);

end;

################################################################################
##
#F      IsEqual( <M>, <word1>, <word2>) .   .   . support function tests whether
##                                                  word1=word2 in M
IsEqual:=function(M,u,v)

  local gens,r,s;

  gens:=ShallowCopy(M.generators);
  Append(gens,M.inverses);
  r:=RClassEnumerate(M,[],u);
  s:=RClassEnumerate(M,[],v);

  if Trace(gens,r.Table,v,1)>0 and Trace(gens,s.Table,u,1)>0 and
     Trace(gens,r.Table,u,1)=Trace(gens,r.Table,v,1) then
    return true;
  else
    return false;
  fi;

end;

################################################################################
##
#F      IsHEquivalent( <M>, <word1>, <word2>).  . support function tests whether
##                                                  word1Hword2 in M

#Note that this is mathematically dodgy as it claims that two factored R-classes
#are identical if they are isomorphic.

IsHEquivalent:=function(M, word1, word2)
```

```
  local gens,r,s;

  gens:=ShallowCopy(M.generators);
  Append(gens,M.inverses);
  r:=ROverHEnumerate(M,[],word1);
  s:=ROverHEnumerate(M,[],word2);

  if Trace(gens,r.Table,word2,1)>0 and Trace(gens,s.Table,word1,1)>0 and
     Trace(gens,r.Table,word1,1)=Trace(gens,r.Table,word2,1) then
    return true;
  else
    return false;
  fi;

end;

#############################################################################
##
#F      IsREquivalent( <M>, <word1>, <word2>).  . support function tests whether
##                                               word1Rword2 in M
IsREquivalent:=function(M, word1, word2)

  local gens,r,s;

  gens:=ShallowCopy(M.generators);
  Append(gens,M.inverses);
  r:=RClassEnumerate(M,[],word1);
  s:=RClassEnumerate(M,[],word2);

  if Trace(gens,r.Table,word2,1)>0 and Trace(gens,s.Table,word1,1)>0 then
    return true;
  else
    return false;
  fi;

end;

#############################################################################
##
#F      IsLEquivalent( <M>, <word1>, <word2>).  . support function tests whether
##                                               word1Lword2 in M
IsLEquivalent:=function(M, word1, word2)

  return IsREquivalent(M,Inverted(Concatenation(M.generators,M.inverses),word1),
                       Inverted(Concatenation(M.generators,M.inverses),word2));

end;
```