Computing with Congruences on Finite 0-Simple Semigroups

Michael Torpey

19th May 2014

Abstract

Congruences are a core topic of interest in semigroup theory: the homomorphic images of a semigroup are described uniquely by its congruences (and the quotients they define) in much the same way that a group's homomorphic images are described by its normal subgroups. This fundamental indicator of a semigroup's structure, therefore, is one of the first things that should be considered when attempting to describe a semigroup computationally. To be able to represent a congruence, and to compute interesting information about it, tends to be computationally difficult with the methods that have so far been described in theory and implemented in computer algebra systems such as GAP. In particular, methods for use with semigroup congruences are very limited.

In this paper we present a new way of representing semigroup congruences on one class of semigroups – the finite 0-simple semigroups – and we present a wide range of functions which can be applied to them to find interesting properties quickly. These methods are largely based on existing theory, but improve on current implementations of congruences, for example the GAP system's representation by a set of generating pairs. We also present methods for converting each way between these two congruence representations, and we present a method to find all the congruences of a given semigroup.

Finally we extend the theory to provide an algorithm which decides whether a semigroup is *congruence-free*. Whereas the other methods apply only to finite 0-simple semigroups, this method applies to all finite semigroups.

Contents

Chapter 1

Introduction

1.1 Motivation

Congruences are a core topic of interest in semigroup theory: the homomorphic images of a semigroup are described uniquely by its congruences in much the same way that a group's homomorphic images are described by its normal subgroups. This is known as the First Isomorphism Theorem: Theorem 1.5.2 in [1, p.23] for semigroups, and Theorem 2.9 in [2, p.16] for groups. Hence congruences are a fundamental indicator of the structure of a semigroup, and are therefore one of the first things that should be considered when attempting to describe a semigroup computationally. However, to be able to represent a congruence, and to compute interesting information about it, tends to be computationally difficult with the methods that have so far been described in theory and implemented in computer algebra systems such as GAP.[4] Compared to the level of research which has gone into computational group theory, computational semigroup theory is in some ways in its infancy, without the iterations of review and improvement which group algorithms have enjoyed over many years. See for example [2], a large collection of group algorithms to which no counterpart exists in semigroup theory. In particular, methods for use with semigroup congruences are very limited.

In this paper we present a new way of representing semigroup congruences on one class of semigroups – the finite 0-simple semigroups – and we present a wide range of functions which can be applied to them to find interesting properties quickly. These methods are largely based on existing theory, but improve on current implementations of congruences, for example the GAP system's representation by a set of generating pairs. We also present methods for converting each way between these two congruence representations, and we present a method to find all the congruences of a given semigroup.

Finally we extend the theory to provide an algorithm which decides whether a semigroup is *congruence-free*. Whereas the other methods apply only to finite 0-simple semigroups, this method applies to all finite semigroups.

1.2 Expected Readership

This text is aimed at readers who have some knowledge of semigroup theory, perhaps having taken a university course at late undergraduate or postgraduate level such as [3]. However, this text is designed to be as accessible as possible to readers with very little semigroup theory, stating as much background theory inside the paper itself as possible, and not relying on external knowledge where it can be avoided. A reference book such as [1] might be useful to such readers, and should be sufficient to understand all of the material.

1.3 GAP Implementation

This theoretical description accompanies a significant body of code which uses the GAP programming language [4] to implement the algorithms which follow. This code has been released under the GNU General Public Licence as part of GAP's **Semigroups** package [5] in the 2.0 release. The full code of the

Semigroups package is attached to this document digitally; the relevant segments written by the author of this paper are in the `gap` directory, as follows:

- All code in `reesmat-cong.gi`
- All code in `univcong.gi`
- The `IsCongruenceFreeSemigroup` method in `properties.gi`

and the declarations in the three corresponding `.gd` files. The functions inside these files implement all the methods described in Chapters 2-5.

1.4 Basic Definitions

First, we need some basic definitions to introduce some of the ideas which will be encountered later on.

Definition 1.1. A **semigroup** is a set S together with a binary operation $* : S \times S \to S$ such that

$$(x * y) * z \quad = \quad x * (y * z)$$

for all $x, y, z \in S$.

Definition 1.2. A **congruence** on a semigroup S is an equivalence relation $\rho \subseteq S \times S$ such that

$$(x, y) \in \rho \quad \text{implies that} \quad (ax, ay), (xa, ya) \in \rho,$$

for all $x, y, a \in S$.

For any semigroup S, the relation $S \times S$, which relates every element of S to every other element of S, fulfills the criteria in Definition 1.2; we call this the **universal congruence**. We also have the relation $\Delta_S = \{(x, x) \mid x \in S\}$, which relates each element only to itself; this also fulfills the criteria, so we call it the **trivial congruence**. Hence every semigroup has at least two congruences.

Definition 1.3. Let S be a semigroup.
A **left ideal** is a non-empty subset $I \subseteq S$ such that $si \in I$ for all $s \in S$ and $i \in I$.
A **right ideal** is a non-empty subset $I \subseteq S$ such that $is \in I$ for all $s \in S$ and $i \in I$.
An **ideal** is a non-empty subset $I \subseteq S$ which is both a left ideal and a right ideal.

Definition 1.4. Let S be a semigroup. S^0 is a semigroup consisting of the elements and multiplication of S together with a zero element 0 such that

$$x0 = 0x = 0$$

for all $x \in S^0$.

Definition 1.5. A **zero** is an element 0 in a semigroup S such that

$$x0 = 0x = 0$$

for all $x \in S$.

Definition 1.6. A semigroup with a zero is **0-simple** if S and $\{0\}$ are its only ideals. [3, p.51]

Definition 1.7. A semigroup with zero is **completely 0-simple** if it is 0-simple and contains left and right ideals which are 0-minimal; i.e. S has a left ideal L which contains no left ideals except L and $\{0\}$, and S has a right ideal R which contains no right ideals except R and $\{0\}$. [3, p.52]

Observe that a finite semigroup is completely 0-simple if and only if it is 0-simple. This paper mainly considers only finite semigroups, so we will usually refer to *finite 0-simple semigroups*, knowing that these must be completely 0-simple. To start with, we define a structure which can be used to describe any finite 0-simple semigroup up to isomorphism:

Definition 1.8. A **Rees 0-matrix semigroup** $\mathcal{M}^0[T; I, \Lambda; P]$ is the set

$$(I \times T \times \Lambda) \cup \{0\}$$

with multiplication given by

$$(i, a, \lambda) \cdot (j, b, \mu) = \begin{cases} (i, ap_{\lambda j}b, \mu) & \text{if } p_{\lambda j} \neq 0, \\ 0 & \text{otherwise,} \end{cases}$$

where

- T is a semigroup,
- I and Λ are index sets,
- P is a $|\Lambda| \times |I|$ matrix with entries $(p_{\lambda i})_{\lambda \in \Lambda, i \in I}$ taken from T^0,
- $0x = x0 = 0$ for all x in the semigroup.

[3, p.52]

Definition 1.9. A matrix P is **regular** if it has no row or column which consists entirely of zeros. That is,

$$\forall i \in I : \exists \lambda \in \Lambda : p_{\lambda i} \neq 0,$$
$$\forall \lambda \in \Lambda : \exists i \in I : p_{\lambda i} \neq 0.$$

[1, p.70]

Theorem 1.10. (The Rees Theorem) *Every completely 0-simple semigroup is isomorphic to a Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$, where G is a group and P is regular. Conversely, every such Rees 0-matrix semigroup is completely 0-simple.*

Proof. Theorem 3.2.3 in [1, p.72-75]. □

Now we can describe any finite 0-simple semigroup as its isomorphic Rees 0-matrix semigroup. Hence we can restrict our further investigations just to this type of semigroup. From now on, when we see a Rees 0-matrix semigroup over G, we will assume that G is a group. Hence "a finite 0-simple Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$" refers to a finite 0-simple Rees 0-matrix semigroup over a group, with a regular matrix P.

1.5 Congruences and Linked Triples

Next we consider the congruences of a finite 0-simple semigroup.

Definition 1.11. For a finite 0-simple Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$, a **linked triple** is a triple

$$(N, \mathcal{S}, \mathcal{T})$$

consisting of a normal subgroup $N \trianglelefteq G$, an equivalence relation $\mathcal{S}$ on I and an equivalence relation $\mathcal{T}$ on Λ, such that the following are satisfied:

1. $\mathcal{S} \subseteq \varepsilon_I$, where $\varepsilon_I = \{(i, j) \in I \times I \,|\, \forall \lambda \in \Lambda : p_{\lambda i} = 0 \iff p_{\lambda j} = 0\}$,
2. $\mathcal{T} \subseteq \varepsilon_\Lambda$, where $\varepsilon_\Lambda = \{(\lambda, \mu) \in \Lambda \times \Lambda \,|\, \forall i \in I : p_{\lambda i} = 0 \iff p_{\mu i} = 0\}$,
3. For all $i, j \in I$ and $\lambda, \mu \in \Lambda$ such that $p_{\lambda i}, p_{\lambda j}, p_{\mu i}, p_{\mu j} \neq 0$ and either $(i, j) \in \mathcal{S}$ or $(\lambda, \mu) \in \mathcal{T}$, we have that $q_{\lambda \mu i j} \in N$, where
$$q_{\lambda \mu i j} = p_{\lambda i} p_{\mu i}^{-1} p_{\mu j} p_{\lambda j}^{-1}.$$

[1, p.86]

We can associate the linked triples of a finite 0-simple semigroup with its non-universal congruences, using a function Γ defined as follows.

Theorem 1.12. *For a finite 0-simple Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$, there exists a mapping Γ from the non-universal congruences ρ onto the linked triples $(N, \mathcal{S}, \mathcal{T})$.*

Proof. This proof and these definitions are adapted from material in [1, p.83-85].

First we will define the mapping Γ, and then we will prove that the triples in its image are all linked.

Let $S = \mathcal{M}^0[G; I, \Lambda; P]$ be a finite 0-simple Rees 0-matrix semigroup. Hence by the Rees Theorem (1.10) G is a group and P is regular.

Let ρ be a congruence on S, and consider the zero element. If $(x, 0) \in \rho$, then $(xy, 0y) = (xy, 0) \in \rho$ and $(yx, y0) = (yx, 0) \in \rho$ for any $y \in S$. Hence the congruence class $0/\rho$ is an ideal of S. Since S is 0-simple, $0/\rho$ must be equal to S or $\{0\}$. In the first case, ρ is the universal congruence $S \times S$. The second case covers all the *non-universal* congruences, and so for all these congruences 0 is related only to itself.

Recall the relations ε_I and ε_Λ as in Definition 1.11: ε_I relates the columns i and j if and only if those columns contain zero entries in the same rows. Similarly, ε_Λ relates the rows λ and μ if and only if those rows contain zero entries in the same columns.

Now let ρ be a *non-universal* congruence on S. We define a relation $\mathcal{S}_\rho \subseteq I \times I$ by the rule that $(i, j) \in \mathcal{S}_\rho$ if and only if

- $(i, j) \in \varepsilon_I$, and
- $(i, p_{\lambda i}^{-1}, \lambda) \; \rho \; (j, p_{\lambda j}^{-1}, \lambda)$

for all $\lambda \in \Lambda$ such that $p_{\lambda i} \neq 0$ (and hence by ε_I, $p_{\lambda j} \neq 0$).

Since ρ and ε_I are reflexive, symmetric and transitive, we can see that $\mathcal{S}_\rho$ is also reflexive, symmetric and transitive. Hence $\mathcal{S}_\rho$ is an equivalence relation on I.

Similarly, we define a relation $\mathcal{T}_\rho \subseteq \Lambda \times \Lambda$ by the rule that $(\lambda, \mu) \in \mathcal{T}_\rho$ if and only if

- $(\lambda, \mu) \in \varepsilon_\Lambda$, and
- $(i, p_{\lambda i}^{-1}, \lambda) \; \rho \; (i, p_{\mu i}^{-1}, \mu)$

for all $i \in I$ such that $p_{\lambda i} \neq 0$ (and hence by ε_I, $p_{\mu i} \neq 0$). And similarly, $\mathcal{T}_\rho$ is an equivalence relation on Λ.

Finally, we want to find a normal subgroup $N_\rho \trianglelefteq G$ to complete the triple. First we need to choose a matrix entry in P which is not equal to 0. Since P is regular, the top row, which for this proof we will call 1_Λ, contains a non-zero entry. Label as 1_I the first (furthest left) column such that $p_{1_\Lambda 1_I} \neq 0$. Now let

$$N_\rho = \{a \in G \mid (1_I, a, 1_\Lambda) \; \rho \; (1_I, 1_G, 1_\Lambda),$$

where 1_G is the identity in the group G. For convenience we may abbreviate 1_I and 1_Λ to 1 when their use is clear.

Now we are ready to define Γ as the function

$$\Gamma : \rho \mapsto (N_\rho, \mathcal{S}_\rho, \mathcal{T}_\rho).$$

which maps any non-universal congruence onto a triple with the entries defined above. We now need to show that this is a *linked triple* as in Definition 1.11.

Since by reflexivity $(1, 1_G, 1) \; \rho \; (1, 1_G, 1)$, we have that $1_G \in N_\rho$, so $N_\rho \neq \varnothing$. We now wish to show that N_ρ is a subgroup of G.

Let $a, b \in N_\rho$, so that

$$(1, a, 1) \; \rho \; (1, 1_G, 1) \quad \text{and} \quad (1, b, 1) \; \rho \; (1, 1_G, 1).$$

Hence, since ρ is a congruence,

$$(1, a, 1)(1, p_{11}^{-2}, 1)(1, b, 1) \; \rho \; (1, 1_G, 1)(1, p_{11}^{-2}, 1)(1, 1_G, 1),$$

which implies
$$(1, ab, 1)\ \rho\ (1, 1_G, 1),$$
and so $ab \in N_\rho$.

Now let $a \in N_\rho$, so that
$$(1, a, 1)(1, p_{11}^{-1}a^{-1}, 1)\ \rho\ (1, 1_G, 1)(1, p_{11}^{-1}a^{-1}, 1),$$
and so
$$(1, 1_G, 1)\ \rho\ (1, a^{-1}, 1).$$
Hence $a^{-1} \in N_\rho$. Now we know that N_ρ is closed under multiplication and inverses, so $N_\rho \leq G$ as required. Next we wish to show that N_ρ is normal.

Let $a \in N_\rho, g \in G$. So
$$(1, g^{-1}p_{11}^{-1}, 1)(1, a, 1)(1, p_{11}^{-1}g, 1)\ \rho\ (1, g^{-1}p_{11}^{-1}, 1)(1, 1_G, 1)(1, p_{11}^{-1}g, 1),$$
that is,
$$(1, g^{-1}ag, 1)\ \rho\ (1, 1_G, 1),$$
and so $g^{-1}ag \in N_\rho$, and $N_\rho \trianglelefteq G$.

So for a non-universal congruence ρ, we have a triple $(N_\rho, \mathcal{S}_\rho, \mathcal{T}_\rho)$ consisting of a normal subgroup $N_\rho \trianglelefteq G$, a column equivalence $\mathcal{S}_\rho \subseteq I \times I$, and a row equivalence $\mathcal{T}_\rho \subseteq \Lambda \times \Lambda$. To prove that this triple is *linked* in the sense of Definition 1.11, we just need to prove condition (3).

Let $i, j \in I$ and $\lambda, \mu \in \Lambda$ such that $p_{\lambda i}, p_{\lambda j}, p_{\mu i}, p_{\mu j} \neq 0$, and require that either $(i, j) \in \mathcal{S}_\rho$ $(\lambda, \mu) \in \mathcal{T}_\rho$. Without loss of generality, suppose that $(i, j) \in \mathcal{S}_\rho$ (the case for $(\lambda, \mu) \in \mathcal{T}_\rho$ is similar). Now by the definition of $\mathcal{S}_\rho$,
$$(i, p_{\mu i}^{-1}, \mu)\ \rho\ (j, p_{\mu j}^{-1}, \mu),$$
and so
$$(1, 1_G, \lambda)(i, p_{\mu i}^{-1}, \mu)(j, p_{\lambda j}^{-1}, 1)\ \rho\ (1, 1_G, \lambda)(j, p_{\mu j}^{-1}, \mu)(j, p_{\lambda j}^{-1}, 1),$$
which implies that
$$(1, p_{\lambda i}p_{\mu i}^{-1}p_{\mu j}p_{\lambda j}^{-1}, 1)\ \rho\ (1, p_{\lambda j}p_{\mu j}^{-1}p_{\mu j}p_{\lambda j}^{-1}, 1),$$
that is,
$$(1, q_{\lambda\mu ij}, 1)\ \rho\ (1, 1_G, 1).$$
Hence $q_{\lambda\mu ij} \in N$, and so $(N_\rho, \mathcal{S}_\rho, \mathcal{T}_\rho)$ is *linked.* □

We now know that Γ is a function which maps non-universal congruences onto linked triples. In order to use congruences and linked triples interchangeably, we need Γ to be a bijection. We prove this in Theorems 1.15 and 1.16 shortly, but first we need to establish a lemma.

Lemma 1.13. *Let ρ be a congruence on a finite 0-simple Rees 0-matrix semigroup over a group.*
$$(1, a, 1)\ \rho\ (1, b, 1) \quad \textit{if and only if} \quad ab^{-1} \in N_\rho$$
for all $a, b \in G$. [1, p.85]

Proof. Let $(1, a, 1)\ \rho\ (1, b, 1)$. Hence
$$(1, a, 1)(1, p_{11}^{-1}b^{-1}, 1)\ \rho\ (1, b, 1)(1, p_{11}^{-1}b^{-1}, 1),$$
and so $(1, ab^{-1}, 1)\ \rho\ (1, 1_G, 1)$, and therefore $ab^{-1} \in N_\rho$. Each of these implications applies in both directions, so the converse is true. □

Lemma 1.14. *If $(i, a, \lambda)\ \rho\ (j, b, \mu)$ then $(i, j) \in \mathcal{S}_\rho$ and $(\lambda, \mu) \in \mathcal{T}_\rho$.*

Proof. This proof is adapted from Lemma 3.5.3 in [1, p.86].

By definition (as in Theorem 1.12), $(i,j) \in \mathcal{S}_\rho$ if and only if $(i,j) \in \varepsilon_I$ and $(i, p_{\xi i}^{-1}, \xi)\ \rho\ (j, p_{\xi j}^{-1}, \xi)$ for every $\xi \in \Lambda$ such that $p_{\xi i} \neq 0$.

To show that $(i,j) \in \varepsilon_I$, choose $\xi \in \Lambda$ such that $p_{\xi i} = 0$. We have

$$0 = (1, 1_G, \xi)(i, a, \lambda)\ \rho\ (1, 1_G, \xi)(j, b, \mu),$$

and since $|0/\rho| = 1$, we have $(1, 1_G, \xi)(j, b, \mu) = 0$, which implies that $p_{\xi j} = 0$. So $p_{\xi i} = 0$ implies $p_{\xi j} = 0$, and by similar reasoning $p_{\xi j} = 0$ implies $p_{\xi i} = 0$. Hence $(i,j) \in \varepsilon_I$.

Now, to show the other part of the definition, we present the following chain of ρ-related pairs, where $x \in I$ and $\xi \in \Lambda$ such that $p_{\xi i}, p_{\lambda x} \neq 0$:

$$(i, a, \lambda)\ \rho\ (j, b, \mu)$$

$$\begin{gathered}(1, 1_G, \xi)(i, a, \lambda)\ \rho\ (1, 1_G, \xi)(j, b, \mu)\\ (1, 1_G, \xi)(i, a, \lambda)(x, p_{11}, 1)\ \rho\ (1, 1_G, \xi)(j, b, \mu)(x, p_{11}, 1)\\ (1, p_{\xi i} a p_{\lambda x} p_{11}, 1)\ \rho\ (1, p_{\xi j} b p_{\mu x} p_{11}, 1)\end{gathered}$$

Let this be written as $(1, c, 1)\ \rho\ (1, d, 1)$, and from Lemma 1.13 we have $cd^{-1} \in N_\rho$. Since N_ρ is normal, we have $c^{-1}(cd^{-1})^{-1}c = c^{-1}d \in N_\rho$, and so again by Lemma 1.13,

$$(1, c^{-1}, 1)\ \rho\ (1, d^{-1}, 1)$$

$$\begin{gathered}(x, 1_G, 1)(1, c^{-1}, 1)(1, p_{11}^{-1}, \xi)\ \rho\ (x, 1_G, 1)(1, d^{-1}, 1)(1, p_{11}^{-1}, \xi)\\ (i, a, \lambda)(x, 1_G, 1)(1, c^{-1}, 1)(1, p_{11}^{-1}, \xi)\ \rho\ (j, b, \mu)(x, 1_G, 1)(1, d^{-1}, 1)(1, p_{11}^{-1}, \xi)\\ (i, p_{\xi i}^{-1}, \xi)\ \rho\ (j, p_{xij}^{-1}, \xi)\end{gathered}$$

and we have the result required. Hence $(i,j) \in \mathcal{S}_\rho$. A similar argument gives the stated result for $\mathcal{T}_\rho$. $\square$

Theorem 1.15. *The function Γ is injective.*

Proof. We want to prove that for two non-universal congruences ρ and σ, $\Gamma(\rho) = \Gamma(\sigma)$ if and only if $\rho = \sigma$.

Clearly $\rho = \sigma$ implies that $\Gamma(\rho) = \Gamma(\sigma)$. Conversely, let ρ and σ be non-universal congruences on S with

$$\Gamma(\rho) = \Gamma(\sigma) = (N, \mathcal{S}, \mathcal{T}),$$

and let

$$(i, a, \lambda)\ \rho\ (j, b, \mu).$$

By Theorem 1.14, $(i,j) \in \mathcal{S}$ and $(\lambda, \mu) \in \mathcal{T}$. Now choose $k \in I, \nu \in \Lambda$ as follows: k is the first column such that $p_{\lambda k} \neq 0$ (and so $p_{\mu k} \neq 0$), and ν is the first row such that $p_{\nu i} \neq 0$ (and so $p_{\nu j} \neq 0$). We have

$$(1, 1_G, \nu)(i, a, \lambda)(k, 1_G, 1)\ \rho\ (1, 1_G, \nu)(j, b, \mu)(k, 1_G, 1),$$

that is, $(1, p_{\nu i} a p_{\lambda k}, 1)\ \rho\ (1, p_{\nu j} b p_{\mu k}, 1)$. By Lemma 1.13, $p_{\nu i} a p_{\lambda k} (p_{\nu j} b p_{\mu k})^{-1} \in N$, and so we can go back to

$$(1, p_{\nu i} a p_{\lambda k}, 1)\ \sigma\ (1, p_{\nu j} b p_{\mu k}, 1).$$

Now, since $(i,j) \in \mathcal{S}$ we have the following:

$$(i, p_{\nu i}^{-1}, \nu)\ \sigma\ (j, p_{\nu j}^{-1}, \nu),$$

$$\begin{gathered}(i, p_{\nu i}^{-1}, \nu)(i, p_{\nu i}^{-1} p_{11}^{-1}, 1)\ \sigma\ (j, p_{\nu j}^{-1}, \nu)(i, p_{\nu i}^{-1} p_{11}^{-1}, 1),\\ (i, p_{\nu i}^{-1} p_{11}^{-1}, 1)\ \sigma\ (j, p_{\nu j}^{-1} p_{11}^{-1}, 1),\end{gathered}$$

and by a similar argument, since $(\lambda, \mu) \in \mathcal{T}$,

$$(1, p_{11}^{-1} p_{\lambda k}^{-1}, \lambda)\ \sigma\ (1, p_{11}^{-1} p_{\mu k}^{-1}, \mu).$$

Now, using the three related pairs we have established, and using the fact that σ is a congruence, we have

$$(i, p_{\nu i}^{-1}p_{11}^{-1}, 1)(1, p_{\nu i}ap_{\lambda k}, 1)(1, p_{11}^{-1}p_{\lambda k}^{-1}, \lambda) \ \sigma \ (j, p_{\nu j}^{-1}p_{11}^{-1}, 1)(1, p_{\nu j}bp_{\mu k}, 1)(1, p_{11}^{-1}p_{\mu k}^{-1}, \mu),$$

which is to say that $(i, a, \lambda) \ \sigma \ (j, b, \mu)$. Hence $\rho = \sigma$, and Γ is injective. [1, p.87]

□

Theorem 1.16. *The function Γ is surjective.*

Proof. This proof is adapted from Lemma 3.5.6 in [1, p.87-90].

We need to show that for each linked triple $(N, \mathcal{S}, \mathcal{T})$ there exists some congruence ρ such that $\Gamma(\rho) = (N, \mathcal{S}, \mathcal{T})$. Let $(N, \mathcal{S}, \mathcal{T})$ be a linked triple, and define a relation ρ such that two non-zero elements (i, a, λ) and (j, b, μ) are ρ-related if and only if

1. $(i, j) \in \mathcal{S}$;
2. $(\lambda, \mu) \in \mathcal{T}$;
3. $p_{\xi i}ap_{\lambda x}(p_{\xi j}bp_{\mu x})^{-1} \in N$ for some $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi j}, p_{\lambda x}, p_{\mu x} \neq 0$;

and 0 is related only to itself.

We wish to establish that ρ is a non-universal congruence, and that $\Gamma(\rho) = (N, \mathcal{S}, \mathcal{T})$. But first we establish a lemma to help us prove this.

Lemma 1.17. *If condition 3 above holds for SOME $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi j}, p_{\lambda x}, p_{\mu x} \neq 0$, then it must hold for ALL $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi j}, p_{\lambda x}, p_{\mu x} \neq 0$.*

Proof. This proof comes from Lemma 3.5.7 in [1, p.88].

Let $x \in I, \xi \in \Lambda$ be such that $p_{\xi i}, p_{\xi j}, p_{\lambda x}, p_{\mu x} \neq 0$ and $p_{\xi i}ap_{\lambda x}(p_{\xi j}bp_{\mu x})^{-1} \in N$.

Also let $y \in I, \eta \in \Lambda$ be such that $p_{\eta i}, p_{\eta j}, p_{\lambda y}, p_{\mu y} \neq 0$. We wish to show that

$$p_{\eta i}ap_{\lambda y}(p_{\eta j}bp_{\mu y})^{-1} \in N.$$

First, for any $c, d \in G$ such that $cd \in N$, we know that $dc \in N$ (since $dc = c^{-1}(cd)c$). Hence, since we have $p_{\xi i}ap_{\lambda x}(p_{\xi j}bp_{\mu x})^{-1} \in N$, we also have

$$p_{\lambda x}p_{\mu x}^{-1}b^{-1}p_{\xi j}^{-1}p_{\xi i}a \in N.$$

Note that since $(\lambda, \mu) \in \mathcal{T}$, $(N, \mathcal{S}, \mathcal{T})$ is a linked triple, and $p_{\lambda x}, p_{\lambda y}, p_{\mu x}, p_{\mu y} \neq 0$, we have

$$q_{\lambda\mu yx} = p_{\lambda y}p_{\mu y}^{-1}p_{\mu x}p_{\lambda x}^{-1} \in N,$$

and multiplying these two elements gives us

$$p_{\lambda y}p_{\mu y}^{-1}b^{-1}p_{\xi j}^{-1}p_{\xi i}a \in N.$$

Again using the property $cd \mapsto dc$, we have

$$ap_{\lambda y}p_{\mu y}^{-1}b^{-1}p_{\xi j}^{-1}p_{\xi i} \in N.$$

Now since $(j, i) \in \mathcal{S}$, $(N, \mathcal{S}, \mathcal{T})$ is a linked triple, and $p_{\xi i}, p_{\xi j}, p_{\eta i}, p_{\eta j} \neq 0$, we have $q_{\xi\eta ji} \in N$, and so

$$p_{\xi i}^{-1}q_{\xi\eta ji}p_{\xi i}^{-1} = p_{\xi i}^{-1}p_{\xi j}p_{\eta j}^{-1}p_{\eta i} \in N.$$

The product of these two elements is $ap_{\lambda y}p_{\mu y}^{-1}b^{-1}p_{\eta j}^{-1}p_{\eta i} \in N$, which we reorder as before, and collect the inverses, to give

$$p_{\eta i}ap_{\lambda y}(p_{\eta j}bp_{\mu y})^{-1} \in N.$$

□

Now we return to Theorem 1.16, and we wish to show that ρ is a congruence. We can see that ρ is reflexive simply by applying $i = j, a = b, \lambda = \mu$ to the definition. Also, ρ is symmetric, since $\mathcal{S}$ and $\mathcal{T}$ are symmetric, and since $cd^{-1} \in N$ implies that $(cd^{-1})^{-1} = dc^{-1} \in N$.

To show that ρ is transitive, consider three elements $(i, a, \lambda), (j, b, \mu), (k, c, \nu)$ such that

$$(i, a, \lambda) \ \rho \ (j, b, \mu) \quad \text{and} \quad (j, b, \mu) \ \rho \ (k, c, \nu).$$

Clearly by the transitivity of $\mathcal{S}$ and $\mathcal{T}$, we have $(i, k) \in \mathcal{S}$ and $(\lambda, \nu) \in \mathcal{T}$. Choose $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi k}, p_{\lambda x}, p_{\nu x} \neq 0$. Since $(i, j) \in \mathcal{S}$ and $(\lambda, \mu) \in \mathcal{T}$, we also have $p_{\xi j}, p_{\mu x} \neq 0$. By Lemma 1.17, these x and ξ are sufficient that

$$p_{\xi i} a p_{\lambda x} (p_{\xi j} b p_{\mu x})^{-1} \in N \quad \text{and} \quad p_{\xi j} a p_{\mu x} (p_{\xi k} c p_{\nu x})^{-1} \in N,$$

and these multiply to give

$$p_{\xi i} a p_{\lambda x} (p_{\xi k} c p_{\nu x})^{-1} \in N,$$

and therefore $(i, a, \lambda) \ \rho \ (k, c, \nu)$, so ρ is transitive.

To show that ρ is a left congruence, again consider three elements $(i, a, \lambda), (j, b, \mu), (k, c, \nu)$ such that $(i, a, \lambda) \ \rho \ (j, b, \mu)$. We need to prove that $(k, c, \nu)(i, a, \lambda) \ \rho \ (k, c, \nu)(j, b, \mu)$.

Since $(i, j) \in \mathcal{S}$, $p_{\nu i}$ and $p_{\nu j}$ are either both zero or both non-zero. If they are both zero, then

$$(k, c, \nu)(i, a, \lambda) = (k, c, \nu)(j, b, \mu) = 0,$$

and so the two products are certainly ρ-related. If they are both non-zero, then we need to show that

$$(k, c p_{\nu i} a, \lambda) \ \rho \ (k, c p_{\nu j} b, \mu).$$

Certainly we have that $(k, k) \in \mathcal{S}$ and $(\lambda, \mu) \in \mathcal{T}$. Hence we just need that

$$p_{\xi k} c p_{\nu i} a p_{\lambda x} (p_{\xi k} c p_{\nu j} b p_{\mu x})^{-1} \in N$$

for some $x \in I$ and $\xi \in \Lambda$ with appropriate non-zero entries.

We have that $(i, a, \lambda) \ \rho \ (j, b, \mu)$ and therefore

$$p_{\xi i} a p_{\lambda x} (p_{\xi j} b p_{\mu x})^{-1} \in N$$

for every $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi j}, p_{\lambda x}, p_{\mu x} \neq 0$. Since in this case $p_{\nu i}, p_{\nu j} \neq 0$, we set $\xi = \nu$ and we immediately have

$$p_{\nu i} a p_{\lambda x} (p_{\nu j} b p_{\mu x})^{-1} \in N$$

for any $x \in I$ with $p_{\lambda x}, p_{\mu x} \neq 0$. If we conjugate this formula by the element $c^{-1} p_{\xi k}^{-1}$, we get the formula we wanted, and ρ is a left congruence. By a similar argument, ρ is a right congruence, and so we have the result that ρ is a two-sided congruence.

Now that we know ρ is a non-universal congruence, we need only to show that $\Gamma(\rho) = (N, \mathcal{S}, \mathcal{T})$, and we will know that Γ is surjective.

First we show that $\mathcal{S} = \mathcal{S}_\rho$. If $(i, j) \in \mathcal{S}_\rho$ then $(i, p_{\lambda i}^{-1}, \lambda) \ \rho \ (j, p_{\lambda j}^{-1}, \lambda)$ for every $\lambda \in \Lambda$ such that $p_{\lambda i} \neq 0$; hence $(i, j) \in \mathcal{S}$. Conversely, if $(i, j) \in \mathcal{S}$ then for all $\lambda \in \Lambda$ such that $p_{\lambda i} \neq 0$,

$$q_{\xi \lambda i j} = p_{\xi i} p_{\lambda i}^{-1} (p_{\lambda x} p_{\lambda x}^{-1}) (p_{\lambda j}^{-1})^{-1} p_{\xi j}^{-1} \in N$$

for all $x \in I, \xi \in \Lambda$ such that $p_{\xi i}, p_{\xi j}, p_{\lambda x} \neq 0$. Hence $(i, p_{\lambda i}, \lambda) \ \rho \ (j, p_{\lambda j}, \lambda)$ and so by definition, $(i, j) \in \mathcal{S}_\rho$. Hence $\mathcal{S} = \mathcal{S}_\rho$. By a similar argument, $\mathcal{T} = \mathcal{T}_\rho$.

Finally we wish to show that $N = N_\rho$. By definition, $a \in N_\rho$ if and only if $(1, a, 1) \ \rho \ (1, 1_G, 1)$. This is the case if and only if (by Lemma 1.17)

$$p_{\xi 1} a p_{1x} (p_{\xi 1} 1_G p_{1x})^{-1} \in N$$

for some $x \in I$ and $\xi \in \Lambda$ such that $p_{\xi 1}, p_{1x} \neq 0$. This is true if and only if $a \in N$. Hence we have $N = N_\rho$. $\square$

Now we have proved that the non-universal congruences of a finite 0-simple semigroup correspond bijectively to its linked triples, and we have come across some results which allow us to determine certain properties of the congruence using its linked triple. In Chapter 2, we will describe how this theory can be used to represent congruences computationally, and the advantages this gives us in finding low-complexity algorithms which can be used to describe the properties of these congruences.

While in this paper we are concentrating on finite 0-simple semigroups, it should be noted that a very similar representation exists for finite *simple* semigroups (Rees matrix semigroups), and the functions described operate in almost exactly the same way, except for the removal of complications related to the zero element. In this system, the universal congruence can also be represented by a linked triple, namely $(G, I \times I, \Lambda \times \Lambda)$. [1, p.90-91]

Since the theory is so similar, we will not explain the finite simple case in any more detail, and instead we will restrict our arguments to semigroups *with* zero, apart from a brief excursion to Rees matrix semigroups in Theorem 5.4. However, a future implementation of this theory in GAP could be useful, and would be a logical extension of this project.

Chapter 2

Semigroup Congruences By Linked Triple

Since the linked triples of a Rees 0-matrix semigroup are in bijective correspondence with its non-universal congruences, we can describe a congruence ρ computationally by its associated linked triple $\Gamma(\rho)$, rather than by storing a set of pairs or a partition of the semigroup. Using this representation we can compute some important properties of the congruence much faster than by these other representations.

In this section, we explain several of the algorithms for computing with these objects, as implemented in GAP by the author of this paper, and where appropriate we provide pseudo-code or well-commented GAP source code.

2.1 Describing Linked Triples Computationally

In order to store a semigroup congruence computationally by its linked triple, we need a function which takes four arguments:

- a finite 0-simple Rees 0-matrix semigroup S,
- a group N,
- a relation $\mathcal{S}$, and
- a relation $\mathcal{T}$,

and returns an object which can be used as a semigroup congruence, but which is stored internally as a linked triple $(N, \mathcal{S}, \mathcal{T})$.

The function will also need to carry out checks to ensure that these arguments describe a valid linked triple. If G is the underlying group in the Rees 0-matrix semigroup $S = \mathcal{M}^0[G; I, \Lambda; P]$, then

- N must be a normal subgroup of G,
- $\mathcal{S}$ must be an equivalence relation on the columns I, and
- $\mathcal{T}$ must be an equivalence relation on the rows Λ.

The relations $\mathcal{S}$ and $\mathcal{T}$, which we traditionally view as sets of pairs in $I \times I$ and $\Lambda \times \Lambda$, can be stored more concisely as partitions of the set I or Λ: lists of lists of integers describing the equivalence classes. For example, if there are 6 columns, and if $\mathcal{S}$ is the equivalence relation with classes $\{1, 3\}$, $\{2, 4, 5\}$ and $\{6\}$, then it might be represented as the list of lists

$$\big[[1, 3], [2, 4, 5], [6]\big],$$

to use square-bracket notation for a list.

This function checks that these conditions are satisfied, then calls the function IsLinkedTriple (see Section 2.3) to check that $(N, \mathcal{S}, \mathcal{T})$ is linked in the sense of Definition 1.11, and then creates the object.

2.2 Finding the Congruences of a Semigroup

Given a finite 0-simple Rees 0-matrix semigroup S over a group G, we wish for a function that returns a list of all the congruences on S, including all non-universal congruences described by their linked triples, and a special object for the universal congruence.

The basic operation of this function is not complicated: find all triples of the form $(N, \mathcal{S}, \mathcal{T})$ and if they are linked, add them to a list. Then take the congruences $\Gamma^{-1}\big((N, \mathcal{S}, \mathcal{T})\big)$ and add the universal congruence. This algorithm is shown in the following pseudo-code:

```
procedure CongruencesOfSemigroup($S$)
    $C := \varnothing$
    for all $N \trianglelefteq G$ do
        for all $\mathcal{S} \subseteq \varepsilon_I$ do
            for all $\mathcal{T} \subseteq \varepsilon_\Lambda$ do
                if IsLinkedTriple($S, N, \mathcal{S}, \mathcal{T}$) then
                    Add $(N, \mathcal{S}, \mathcal{T})$ to $C$
                end if
            end for
        end for
    end for
    Replace each triple in $C$ with its congruence
    Add UniversalSemigroupCongruence($S$) to $C$
    return $C$
end procedure
```

This algorithm is simple in itself. However, there is also the complication of calculating the normal subgroups N, as well as ε_I and ε_Λ and their subsets.

To calculate all the normal subgroups $N \trianglelefteq G$, the attached implementation uses the `NormalSubgroups` method currently implemented in GAP [4], which uses the method described in [6].

The relation $\varepsilon_I \subseteq I \times I$ is defined as the equivalence relation on the columns of the semigroup which relates columns with zero entries in exactly the same places. That is,

$$(i, j) \in \varepsilon_I \quad \text{if and only if} \quad \{\lambda \in \Lambda \mid p_{\lambda i} = 0\} = \{\lambda \in \Lambda \mid p_{\lambda j} = 0\}.$$

To calculate this, we use the following algorithm:

```
procedure $\varepsilon_I$
    $\varepsilon_I := \big\{\{1\}, \{2\}, \ldots, \{n\}\big\}$                    ▷ where $n = |I|$
    for $i \in \{1 \ldots n\}$ do
        for $j \in \{i+1 \ldots n\}$ do
            if $(\forall \lambda \in \Lambda) : p_{\lambda i}, p_{\lambda j} \neq 0$ or $p_{\lambda i} = p_{\lambda j} = 0$ then
                AddRelation($\varepsilon_I, i, j$)
            end if
        end for
    end for
    return $\varepsilon_I$
end procedure
```

The row equivalence ε_Λ is calculated similarly. AddRelation takes an equivalence relation and two elements a and b, and modifies the equivalence relation to combine the two classes containing a and b (see Section 2.2.1). To calculate all the subsets $\mathcal{S} \subseteq \varepsilon_I$ and $\mathcal{T} \subseteq \varepsilon_\Lambda$, we use another helper function subpartitions (see Section 2.2.2).

2.2.1 Add a Relation

In order to create the "maximal" column and row relations ε_I and ε_Λ we combine various classes iteratively using the function AddRelation. In this case the equivalence relation ρ is stored as a partition of the integers $\{1 \ldots |I|\}$, and AddRelation(ρ, x, y) modifies the relation ρ so that the classes containing x and y are combined. This algorithm is demonstrated in the following pseudo-code:

```
procedure AddRelation($\rho, x, y$)
    Let $X$ be the first list in $\rho$ containing $x$        ▷ In fact, it is the only such list
    Let $Y$ be the first list in $\rho$ containing $y$
    if $X \neq Y$ then
        $X := X \cup Y$        ▷ Combine the congruence classes of x and y
        $\rho := \rho \setminus Y$        ▷ Remove the redundant class whose elements are now in X
    end if
end procedure
```

To find the complexity of this method, let n be the number of elements in the set. This algorithm searches lists for the elements x and y, making no more than n comparisons for each. Then in comparing the disjoint sets X and Y it needs to make precisely one comparison. Finally, combining the sets X and Y cannot take more than n storage operations. Hence, AddRelation has linear order $O(n)$.

2.2.2 Subpartitions

This function takes a partition of a set $\{1 \ldots n\}$ and returns a list of all the partitions which refine that partition. For example, given the partition

$$\big[[1,3],[2,4,5],[6]\big]$$

subpartitions would return

$$\Big[\ \big[[1,3],[2,4,5],[6]\big],\\ \big[[1,3],[2,4],[5],[6]\big],\\ \big[[1,3],[2,5],[4],[6]\big],\\ \big[[1,3],[2],[4,5],[6]\big],\\ \big[[1,3],[2],[4],[5],[6]\big],\\ \big[[1],[2,4,5],[3],[6]\big],\\ \big[[1],[2,4],[3],[5],[6]\big],\\ \big[[1],[2,5],[3],[4],[6]\big],\\ \big[[1],[2],[3],[4,5],[6]\big],\\ \big[[1],[2],[3],[4],[5],[6]\big]\ \Big],$$

the list of all "subpartitions" of the argument. The algorithm for doing this is best demonstrated by the following high-level GAP implementation:

```
subpartitions := function(part)
  local l;
  # Replace each class with a list of all partitions of that class
  l := List(part, PartitionsSet);
  # Produce all the combinations of partitions of classes
  l := Cartesian(l);
  # Concatenate these lists to produce complete partitions of the set
  l := List(l, Concatenation);
  # Finally sort each of these into the canonical order of its new classes
  l := List(l, SSortedList);
  return l;
end;
```

2.3 Is a Triple Linked?

By definition, a triple $(N, \mathcal{S}, \mathcal{T})$ is *linked* only if it fulfills the three conditions required in Definition 1.11. The IsLinkedTriple function tests these conditions, and returns a boolean value for whether the triple is linked. The three conditions can be tested together with the following algorithm:

```
Require: $S = \mathcal{M}^0[G; I, \Lambda; P]$,   $N \trianglelefteq G$,   $\mathcal{S} \subseteq \varepsilon_I$,   $\mathcal{T} \subseteq \varepsilon_\Lambda$.
  procedure IsLinkedTriple($S, N, \mathcal{S}, \mathcal{T}$)

    ▷ First, the column relation $\mathcal{S}$
    for all equivalence classes $\mathcal{S}_a$ of $\mathcal{S}$ do
      Let $\{i_1, i_2 \ldots i_n\}$ be the elements of $\mathcal{S}_a$
      ▷ Do all columns in $\mathcal{S}_a$ have zero in the same rows?
      for $k = 2$ to $n$ do
        for $\lambda \in \Lambda$ do
          if $p_{\lambda i_1} = 0$ xor $p_{\lambda i_k} = 0$ then
            return false
          end if
        end for
      end for
      ▷ Is $q_{\lambda\mu ij} \in N$ for every $i, j \in \mathcal{S}_a$?
      for $i, j \in \mathcal{S}_a$ do
        for all $\lambda, \mu \in \Lambda$ such that $p_{\lambda i}, p_{\mu i} \neq 0$ do        ▷ and therefore $p_{\lambda j}, p_{\mu j} \neq 0$
          if $p_{\lambda i} p_{\mu i}^{-1} p_{\mu j} p_{\lambda j}^{-1} \notin N$ then
            return false
          end if
        end for
      end for
    end for

    ▷ Second, the row relation $\mathcal{T}$
    for all equivalence classes $\mathcal{T}_a$ of $\mathcal{T}$ do
      Let $\{\lambda_1, \lambda_2 \ldots \lambda_n\}$ be the elements of $\mathcal{T}_a$
      ▷ Do all rows in $\mathcal{T}_a$ have zero in the same columns?
      for $k = 2$ to $n$ do
        for $i \in I$ do
          if $p_{\lambda_1 i} = 0$ xor $p_{\lambda_k i} = 0$ then
            return false
          end if
        end for
      end for
      ▷ Is $q_{\lambda\mu ij} \in N$ for every $\lambda, \mu \in \mathcal{T}_a$?
      for $\lambda, \mu \in \mathcal{T}_a$ do
        for all $i, j \in I$ such that $p_{\lambda i}, p_{\lambda j} \neq 0$ do        ▷ and therefore $p_{\mu i}, p_{\mu j} \neq 0$
          if $p_{\lambda i} p_{\mu i}^{-1} p_{\mu j} p_{\lambda j}^{-1} \notin N$ then
            return false
          end if
        end for
      end for
    end for
  end procedure
```

2.4 Equality

We need a function which compares two congruences ρ_1 and ρ_2 and reports whether they are equal. For two congruences by linked triple, this is simple:

Let ρ_1 have the linked triple $(N_1, \mathcal{S}_1, \mathcal{T}_1)$ and ρ_2 have the linked triple $(N_2, \mathcal{S}_2, \mathcal{T}_2)$. Now $\rho_1 = \rho_2$ if and only if

- ρ_1 and ρ_2 are congruences of the same semigroup S,
- $N_1 = N_2$,
- $\mathcal{S}_1 = \mathcal{S}_2$, and
- $\mathcal{T}_1 = \mathcal{T}_2$.

2.5 Pair Inclusion

We have been storing a congruence ρ by its linked triple. However, for this representation to be useful we must also be able to view it as a set of pairs in $S \times S$, and to this end we provide an inclusion function which tests whether a pair $(x, y) \in S \times S$ is in ρ – that is, whether x is ρ-related to y.

Since the linked triples of a finite 0-simple Rees 0-matrix semigroup over a group describe precisely the *non-universal* congruences, the congruence ρ with linked triple $(N, \mathcal{S}, \mathcal{T})$ contains no pair including zero apart from $(0, 0)$.

If $x, y \neq 0$ then we can write $x = (i, a, \lambda)$ and $y = (j, b, \mu)$. To decide whether $(x, y) \in \rho$, we need to use all three parts of the linked triple, as in the conditions for ρ-congruence in Theorem 1.16. Firstly, it is necessary that

$$(i, j) \in \mathcal{S} \quad \text{and} \quad (\lambda, \mu) \in \mathcal{T}.$$

If either of these conditions is not satisfied, then $(x, y) \notin \rho$. However, if these are both true, then it gives us an important piece of information which is necessary for the next test: since $\mathcal{S} \subseteq \varepsilon_I$, we know that the columns i and j have zero entries in precisely the same rows; and since $\mathcal{T} \subseteq \varepsilon_\Lambda$, we know that the rows λ and μ have zero entries in precisely the same columns.

Let $k \in I$ be any column of the matrix such that $p_{\lambda k} \neq 0$: hence $p_{\mu k} \neq 0$ as well. Similarly let $\nu \in \Lambda$ be any row such that $p_{\nu i} \neq 0$: hence $p_{\nu j} \neq 0$ as well. Recall that P has no rows or columns which are all-zero, so these choices will always be possible. Now we require that the two elements

$$p_{\nu i} \cdot a \cdot p_{\lambda k} \quad \text{and} \quad p_{\nu j} \cdot b \cdot p_{\mu k}$$

must satisfy the condition

$$(p_{\nu i}\ a\ p_{\lambda k}) \cdot (p_{\nu j}\ b\ p_{\mu k})^{-1} \in N.$$

This is the same as deciding whether they are in the same coset of N with respect to G.

If all three of these tests return true (the three conditions in Theorem 1.16) this is sufficient to determine that $(x, y) \in \rho$. This algorithm is shown in the following pseudo-code:

```
Require: $S = \mathcal{M}^0[G; I, \Lambda; P]$
  procedure In($(x, y)$, $\rho$)
      if $x = y$ then                                        ▷ First a special case for 0
          return true
      else if $x = 0$ or $y = 0$ then
          return false
      end if
      Let $(N, \mathcal{S}, \mathcal{T})$ be the linked triple of $\rho$
      Let $x = (i, a, \lambda)$
      Let $y = (j, b, \mu)$
      if $(i, j) \notin \mathcal{S}$ or $(\lambda, \mu) \notin \mathcal{T}$ then
          return false
```

```
    end if
    Let $k$ be the first column in $I$ such that $p_{\lambda k} \neq 0$        ▷ and therefore $p_{\mu k} \neq 0$
    Let $\nu$ be the first row in $\Lambda$ such that $p_{\nu i} \neq 0$        ▷ and therefore $p_{\nu j} \neq 0$
    if $(p_{\nu i}\ a\ p_{\lambda k}) \cdot (p_{\nu j}\ b\ p_{\mu k})^{-1} \in N$ then
        return true
    else
        return false
    end if
end procedure
```

Note that for this algorithm, k and ν need not be the *first* column and row with the specified condition, but could be *any* column and row with that condition. However, the first column and row are reasonable canonical examples, and in Section 3 we will see other methods for which this choice must be deterministic.

2.6 Elements of a Congruence Class

This function takes a congruence ρ and an element $x \in S$ and returns the elements of x/ρ – that is, a list of all elements which are ρ-related to x. If $x = 0$, then it will return only $\{0\}$. Otherwise, we can define x equal to some (i, a, λ), and the function will return all elements (j, b, μ) such that $i\ \mathcal{S}\ j$, $\lambda\ \mathcal{T}\ \mu$, and $(p_{\nu i}\ a\ p_{\lambda k}) \cdot (p_{\nu j}\ b\ p_{\mu k})^{-1} \in N$ (as in Section 2.5 and Theorem 1.16). The GAP system refers to methods of this type by the name `ImagesElm` – the "images" of an element x.

```
Require: $S = \mathcal{M}^0[G; I, \Lambda; P]$
  procedure ImagesElm$(x, \rho)$
    Let $(N, \mathcal{S}, \mathcal{T})$ be the linked triple of $\rho$
    if $x = 0$ then
        return $\{0\}$
    end if
    Let $x = (i, a, \lambda)$
    Let $k$ be the first column in $I$ such that $p_{\lambda k} \neq 0$
    Let $\nu$ be the first row in $\Lambda$ such that $p_{\nu i} \neq 0$
    $R_x := \varnothing$
    for $j \in i/\mathcal{S}$ do
        for $\mu \in \lambda/\mathcal{T}$ do
            for $n \in Na$ do
                $b := p_{\nu j}^{-1}\ n\ p_{\nu i}\ a\ p_{\lambda k}\ p_{\mu k}^{-1}$   ◁
                Add $(j, b, \mu)$ to $R_x$
            end for
        end for
    end for
    return $R_x$
  end procedure
```

In the line marked ◁, we use the fact that we want to find every group element b such that

$$p_{\nu j} b p_{\mu k} \in N p_{\nu i} a p_{\lambda k}.$$

Hence, for every $n \in N$ we calculate what that b must be and add it to the list.

2.7 Join

Given congruences (or equivalences) ρ_1 and ρ_2, the **join** $\rho_1 \vee \rho_2$ is defined as the smallest congruence (or equivalence) ρ' such that $\rho_1 \cup \rho_2 \subseteq \rho'$.

To calculate $\rho_1 \vee \rho_2$ with the linked triple representation, let $(N_1, \mathcal{S}_1, \mathcal{T}_1)$ and $(N_2, \mathcal{S}_2, \mathcal{T}_2)$ be the linked triples of ρ_1 and ρ_2 respectively. By Lemma 3.6.1 in [1, p.91], the linked triple of $\rho_1 \vee \rho_2$ is

$$(N_1 N_2, \mathcal{S}_1 \vee \mathcal{S}_2, \mathcal{T}_1 \vee \mathcal{T}_2).$$

To find $N_1 N_2$, we find a generating set for each of N_1 and N_2. Combining these sets gives a set of generators for $N_1 N_2$. To find $\mathcal{S}_1 \vee \mathcal{S}_2$, we allow the columns in I to be *marked* or *unmarked*. Initially all the columns are unmarked. Now we iterate through the columns: if a column i is marked, we ignore it; if not, then find the union of $i/\mathcal{S}_1$ and $i/\mathcal{S}_2$, and add it to a list of "blocks", the equivalence classes of $\mathcal{S}_1 \vee \mathcal{S}_2$. All of the columns in this new block, we now mark to avoid repeats. $\mathcal{T}_1 \vee \mathcal{T}_2$ is calculated in a similar fashion.

This algorithm is shown in the following pseudo-code:

```
procedure JOIN($\rho_1, \rho_2$)
    Let $(N_1, \mathcal{S}_1, \mathcal{T}_1)$ be the linked triple of $\rho_1$
    Let $(N_2, \mathcal{S}_2, \mathcal{T}_2)$ be the linked triple of $\rho_2$
    Let $X_1$ be a set of generators for $N_1$
    Let $X_2$ be a set of generators for $N_2$
    $N_1 N_2 := \langle X_1, X_2 \rangle$

    $C_I := \varnothing$
    for $i \in I$ do
        if $i$ is marked then
            continue
        end if
        $B_i := i/\mathcal{S}_1 \cup i/\mathcal{S}_2$
        Mark elements in $B_i$
        Add $B_i$ to $C_I$
    end for
    Let $\mathcal{S}_1 \vee \mathcal{S}_2$ be the equivalence with classes $C_I$

    $C_\Lambda := \varnothing$
    for $\lambda \in \Lambda$ do
        if $\lambda$ is marked then
            continue
        end if
        $B_\lambda := \lambda/\mathcal{T}_1 \cup \lambda/\mathcal{T}_2$
        Mark elements in $B_\lambda$
        Add $B_\lambda$ to $C_\Lambda$
    end for
    Let $\mathcal{T}_1 \vee \mathcal{T}_2$ be the equivalence with classes $C_\Lambda$
    Let $\rho$ have linked triple $(N_1 N_2, \mathcal{S}_1 \vee \mathcal{S}_2, \mathcal{T}_1 \vee \mathcal{T}_2)$
    return $\rho$
end procedure
```

2.8 Meet

Lemma 2.1. *Given congruences ρ_1 and ρ_2, the intersection $\rho_1 \cap \rho_2$ is also a congruence.*

Proof. Let $\rho' = \rho_1 \cap \rho_2$, and let $x \ \rho' \ y$ and $u \ \rho' \ v$. Hence $x \ \rho_1 \ y$, $x \ \rho_2 \ y$, $u \ \rho_1 \ v$, and $u \ \rho_2 \ v$. Since ρ_1 and ρ_2 are congruences, $xu \ \rho_1 \ yv$ and $xu \ \rho_2 \ yv$. Hence $xu \ \rho' \ yv$, and so ρ' is a congruence. □

Since $\rho_1 \cap \rho_2$ is the largest congruence which is a subset of both ρ_1 and ρ_2, we call it the **meet** of ρ_1 and ρ_2, and we may write it $\rho_1 \wedge \rho_2$.

To calculate $\rho_1 \wedge \rho_2$ with the linked triple representation, Let $(N_1, \mathcal{S}_1, \mathcal{T}_1)$ and $(N_2, \mathcal{S}_2, \mathcal{T}_2)$ be the linked triples of ρ_1 and ρ_2 respectively. By Lemma 3.6.1 in [1, p.91], the linked triple of $\rho_1 \wedge \rho_2$ is

$$(N_1 \cap N_2, \mathcal{S}_1 \cap \mathcal{S}_2, \mathcal{T}_1 \cap \mathcal{T}_2).$$

To find $N_1 \cap N_2$, the current implementation relies on the GAP method for finding the intersection of two groups.[4] $\mathcal{S}_1 \cap \mathcal{S}_2$ and $\mathcal{T}_1 \cap \mathcal{T}_2$ are found using a method identical to that for the *join* in Section 2.7, except that we find the intersection of the row and column classes instead of the union.

This method is shown in the following pseudo-code:

```
procedure JOIN($\rho_1, \rho_2$)
    Let $(N_1, \mathcal{S}_1, \mathcal{T}_1)$ be the linked triple of $\rho_1$
    Let $(N_2, \mathcal{S}_2, \mathcal{T}_2)$ be the linked triple of $\rho_2$
    Calculate $N_1 \cap N_2$

    $C_I := \varnothing$
    for $i \in I$ do
        if $i$ is marked then
            continue
        end if
        $B_i := i/\mathcal{S}_1 \cap i/\mathcal{S}_2$
        Mark elements in $B_i$
        Add $B_i$ to $C_I$
    end for
    Let $\mathcal{S}_1 \cap \mathcal{S}_2$ be the equivalence with classes $C_I$

    $C_\Lambda := \varnothing$
    for $\lambda \in \Lambda$ do
        if $\lambda$ is marked then
            continue
        end if
        $B_\lambda := \lambda/\mathcal{T}_1 \cap \lambda/\mathcal{T}_2$
        Mark elements in $B_\lambda$
        Add $B_\lambda$ to $C_\Lambda$
    end for
    Let $\mathcal{T}_1 \cap \mathcal{T}_2$ be the equivalence with classes $C_\Lambda$
    Let $\rho$ have linked triple $(N_1 \cap N_2, \mathcal{S}_1 \cap \mathcal{S}_2, \mathcal{T}_1 \cap \mathcal{T}_2)$
    return $\rho$
end procedure
```

2.9 Universal Congruences

As mentioned before, the linked triples of a semigroup S describe its *non-universal* congruences. For any semigroup there also exists the universal semigroup $S \times S$, which must also be considered. For completeness, all methods above which apply to $S \times S$ have been implemented in GAP, though most of them are trivial. Another object is implemented which represents $\mathcal{U}$, the single *universal congruence class* of $S \times S$.

For the universal semigroup congruence $S \times S$ and a semigroup congruence by linked triple ρ, the following results are defined:

- $S \times S = S \times S$,
- $S \times S \neq \rho$,

- $(x, y) \in S \times S$ for all $x, y \in S$,
- ImagesElm$(S \times S, x)$ returns all the elements of S,
- NrCongruenceClasses$(S \times S) = 1$,
- $S \times S \vee S \times S = S \times S$,
- $S \times S \vee \rho = S \times S$,
- $S \times S \wedge S \times S = S \times S$,
- $S \times S \wedge \rho = \rho$,
- EquivalenceClasses$(S \times S) = \{\mathcal{U}\}$,
- EquivalenceClassOfElement$(S \times S, x) = \mathcal{U}$ for all $x \in S$,
- $x \in \mathcal{U}$ for all $x \in S$,
- $\mathcal{U} \cdot \mathcal{U} = \mathcal{U}$ in $S/(S \times S)$,
- $|\mathcal{U}| = |S|$,
- $\mathcal{U} = \mathcal{U}$,
- GeneratingPairsOfCongruence$(S \times S) = \{(x, s_1) \mid \forall x \in S\}$ for some fixed $s_1 \in S$ (see Section 4.2).

Chapter 3

Congruence Classes By Class Triple

Now that we have a way of describing a congruence ρ with its linked triple, and a way of testing whether two elements are ρ-related using that linked triple, we now seek a concise way to describe a congruence class of ρ.

Definition 3.1. Let $S = \mathcal{M}^0[G; I, \Lambda; P]$ be a finite 0-simple Rees 0-matrix semigroup; let ρ be a non-universal congruence on S with linked triple $(N, \mathcal{S}, \mathcal{T})$. A **class triple** of ρ is any triple

$$(Nl, i/\mathcal{S}, \lambda/\mathcal{T}),$$

where

- $i/\mathcal{S}$ is an equivalence class of $\mathcal{S}$;
- $\lambda/\mathcal{T}$ is an equivalence class of $\mathcal{T}$;
- Nl is a coset of N in G.

Theorem 3.2. *The class triples of ρ correspond bijectively to its non-zero congruence classes.*

Proof. Let $x = (i, a, \lambda)$ be an arbitrary non-zero element of S. Let $k \in I$ be the first column in P such that $p_{\lambda k} \neq 0$, and let $\nu \in \Lambda$ be the first row in P such that $p_{\nu i} \neq 0$. We define C_x as the class triple

$$(Nl, i/\mathcal{S}, \lambda/\mathcal{T}),$$

where $l = p_{\nu i} \cdot a \cdot p_{\lambda k}$.

We now need only to show that for $x, y \in S$, $(x, y) \in \rho$ if and only if $C_x = C_y$.

($\Rightarrow$): Let $(x, y) \in \rho$ such that $x = (i, a, \lambda)$ and $y = (j, b, \mu)$. By Lemma 1.14, $i/\mathcal{S} = j/\mathcal{S}$ and $\lambda/\mathcal{T} = \mu/\mathcal{T}$. Now, since $(i, j) \in \varepsilon_I$ and $(\lambda, \mu) \in \varepsilon_\Lambda$, the first non-zero column k will be the same for λ and μ, and the first non-zero row ν will be the same for i and j. Therefore by Theorem 1.16, $p_{\nu i} a p_{\lambda k} (p_{\nu j} b p_{\mu k})^{-1} \in N$, so the two elements $p_{\nu i} \cdot a \cdot p_{\lambda k}$ and $p_{\nu j} \cdot b \cdot p_{\mu k}$ lie in the same coset of N, which is called Nl. Hence $C_x = C_y$.

($\Leftarrow$): Let $x = (i, a, \lambda)$ and $y = (j, b, \mu)$ be chosen such that $C_x = C_y$. Hence $i/\mathcal{S} = j/\mathcal{S}$ and $\lambda/\mathcal{T} = \mu/\mathcal{T}$, so $(i, j) \in \mathcal{S}$ and $(\lambda, \mu) \in \mathcal{T}$. Now since ν and k can be chosen consistently, $p_{\nu i} \cdot a \cdot p_{\lambda k}$ and $p_{\nu j} \cdot b \cdot p_{\mu k}$ lie in the same coset of N, and again we have that $p_{\nu i} a p_{\lambda k} (p_{\nu j} b p_{\mu k})^{-1} \in N$, and so $(x, y) \in \rho$. $\square$

Note that here, unlike the method for pair inclusion (see Section 2.5) we cannot choose k and ν arbitrarily. In order to generate the same class triple from two congruent elements x and y, we must be consistent in which k and ν are used; choosing the first valid column and row from the left and top of the matrix respectively is deterministic, and can be relied upon to be invariant for elements whose columns and rows are $\mathcal{S}$- and $\mathcal{T}$-related.

Throughout this section we will use the notation x/ρ to describe a congruence class of ρ, defining it in terms of a representative element x. However, the use of this notation does not imply that we have calculated what x actually is; x/ρ is simply an arbitrary congruence class, unless x is already defined. A similar rule should be applied to column and row equivalence classes $i/\mathcal{S}$ and $\lambda/\mathcal{T}$, and cosets Nl or Na.

3.1 Describing Class Triples Computationally

In order to store a class triple in a computational algebra system, we need a function which takes four arguments:

- a congruence ρ defined by its linked triple $(N, \mathcal{S}, \mathcal{T})$
- a coset Na,
- a column class $i/\mathcal{S}$, as an integer,
- a row class $\lambda/\mathcal{T}$, as an integer,

and returns a non-zero congruence class of ρ, which we can store internally as the class triple $(Na, i/\mathcal{S}, \lambda/\mathcal{T})$.

The function should first carry out checks to ensure that the supplied arguments make sense. Firstly, it is necessary to check that the coset Na is indeed a coset of N with respect to the Rees 0-matrix semigroup's underlying group G. Secondly, note that since $\mathcal{S}$ and $\mathcal{T}$ are stored as partitions (lists of lists), the classes $i/\mathcal{S}$ and $\lambda/\mathcal{T}$ can be stored simply as an integer which describes the position of that class in the partition; we must therefore test that these numbers are at least 1, and are no larger than the number of classes in their respective equivalence relations.

Once these checks are carried out, a representative for the class should be calculated with CanonicalRepresentative, and it should be stored along with the class triple.

3.2 Finding the Classes of a Congruence

Now that we have a way of describing the classes of a congruence, we give a function that returns a list of all congruence classes of a congruence ρ. This is done by finding all cosets of N, all equivalence classes of $\mathcal{S}$, and all equivalence classes of $\mathcal{T}$, and then combining them into every possible class triple $(Na, i/\mathcal{S}, \lambda/\mathcal{T})$ and applying using those triples to produce congruence class objects. Finally the zero class $0/\rho$ must be added.

```
procedure CongruenceClasses($\rho$)
    Let $(N, \mathcal{S}, \mathcal{T})$ be the linked triple of $\rho$
    $C_\rho := \varnothing$
    for all equivalence classes $i/\mathcal{S}$ of $\mathcal{S}$ do            ▷ where $i \in I$ is a representative of $i/\mathcal{S}$
        for all equivalence classes $\lambda/\mathcal{T}$ of $\mathcal{T}$ do   ▷ where $\lambda \in \Lambda$ is a representative of $\lambda/\mathcal{T}$
            for all cosets $Na$ of $N$ in $G$ do                 ▷ where $a \in G$ is a representative of $Na$
                Add $(Na, i/\mathcal{S}, \lambda/\mathcal{T})$ to $C_\rho$
            end for
        end for
    end for
    Replace each triple in $C_\rho$ by its congruence class
    Add $0/\rho$ to $C_\rho$
    return $C_\rho$
end procedure
```

3.3 Number of Congruence Classes

For a subgroup H of a group G, we define its **index** $|G : H|$ as the number of cosets of H in G. Similarly, for an equivalence relation R on a set X, let us define the index $|X : R|$ to be the number of equivalence classes of R in X.

We define a function NrCongruenceClasses, which takes a congruence ρ and returns $|S : \rho|$, the number of congruence classes ρ has in its semigroup S.

Let ρ have the linked triple $(N, \mathcal{S}, \mathcal{T})$. Since the (non-zero) congruence classes of ρ correspond bijectively to the class triples, and since zero has its own single class,

$$|S : \rho| = \big(|G : N| \cdot |I : \mathcal{S}| \cdot |\Lambda : \mathcal{T}|\big) + 1.$$

For example, if N has 6 cosets in G, $\mathcal{S}$ has 2 equivalence classes in I, and $\mathcal{T}$ has 5 equivalence classes in Λ, then ρ has

$$(6 \times 2 \times 5) + 1 = 61$$

congruence classes in S.

3.4 Congruence Class of an Element

We define a function, CongruenceClassOfElement, which takes as parameters a congruence ρ over a Rees 0-matrix semigroup S specified by a linked triple, and an element $x \in S$.

If $x = 0$, then we simply return $0/\rho$, the class containing only 0. Otherwise, x has the form (i, a, λ), and we can construct its class triple C_x by the method described in the proof of Theorem 3.2. This triple can then be used to describe the congruence class object, using the function defined in Section 3.1, "RZMSCongruenceClassByLinkedTriple".

This algorithm can be seen in the following pseudo-code:

```
procedure CongruenceClassOfElement($(\rho, x)$)
    if $x = 0$ then
        return $0/\rho$
    end if
    Let $(N, \mathcal{S}, \mathcal{T})$ be the linked triple of $\rho$
    Let $x = (i, a, \lambda)$
    Let $k$ be the first column in $I$ such that $p_{\lambda k} \neq 0$
    Let $\nu$ be the first row in $\Lambda$ such that $p_{\nu i} \neq 0$
    $l := p_{\nu i} \cdot a \cdot p_{\lambda k}$
    return RZMSCongruenceClassByLinkedTriple($(\rho, Nl, i/\mathcal{S}, \lambda/\mathcal{T})$)
end procedure
```

3.5 Class Inclusion

This function takes an element $y \in S$ and a congruence class x/ρ, and determines whether $y \in x/\rho$ using class triples. The logic is much the same as determining whether two elements x and y are ρ-related as in Section 2.5, but without using an explicit element x.

```
procedure In($(y, x/\rho)$)
    if $y = 0$ then
        if $x/\rho = 0/\rho$ then
            return true
        else
            return false
        end if
    end if
    Let $y = (j, b, \mu)$
    Let $(Nl, i/\mathcal{S}, \lambda/\mathcal{T})$ be the class triple of $x/\rho$
    Let $k$ be the first column in $I$ such that $p_{\mu k} \neq 0$
    Let $\nu$ be the first row in $\Lambda$ such that $p_{\nu j} \neq 0$
    if $p_{\nu j}\ b\ p_{\mu k} \in Nl$ and $j \in i/\mathcal{S}$ and $\mu \in \lambda/\mathcal{T}$ then
```

```
        return true
    else
        return false
    end if
end procedure
```

3.6 Class Multiplication

If we have a semigroup S with a congruence ρ, we can of course form a new semigroup, the quotient S/ρ, with elements the congruence classes of ρ, and multiplication given by the rule

$$(x/\rho)(y/\rho) = (xy)/\rho$$

for $x, y \in S$. This multiplication can be done computationally simply by multiplying the class representatives x and y and applying the function CongruenceClassOfElement to the product xy (see Section 3.4).

3.7 Size of a Class

This function returns the number of elements in an equivalence class x/ρ.

If x/ρ is the zero class $0/\rho$, then clearly it has size 1. Otherwise it has a class triple $(Nl, i/\mathcal{S}, \lambda/\mathcal{T})$. An element is in the class only if it has a column in $i/\mathcal{S}$ and a row in $\lambda/\mathcal{T}$. It must also have a group element which maps to the coset Nl. Nl has the same size as the normal subgroup N, so

$$|x/\rho| = |N| \cdot |i/\mathcal{S}| \cdot |\lambda/\mathcal{T}|.$$

3.8 Canonical Representative

As mentioned in Section 3.1, we should have a method to calculate a representative of a congruence class using its class triple. CanonicalRepresentative is a function that takes a congruence class specified by a class triple and returns a representative element of that class which can be used to identify the class uniquely.

Consider the congruence class x/ρ. We wish to find some $x_1 \in x/\rho$ which can represent x/ρ canonically. If $x/\rho = 0/\rho$, then we must of course choose $x_1 = 0$. Otherwise, x/ρ has class triple $(Nl, i/\mathcal{S}, \lambda/\mathcal{T})$ and we must choose

$$x_1 = (i_1, a_1, \lambda_1)$$

for some $i_1 \in I, a_1 \in G, \lambda_1 \in \Lambda$.

It is fairly easy to choose a canonical i_1: simply take the furthest left column in $i/\mathcal{S}$. Similarly let λ_1 be the row in $\lambda/\mathcal{T}$ nearest to the top of the matrix. Now, as in Section 2.5, we can define k as the first column such that $p_{\lambda_1 k} \neq 0$, and we can define ν as the first row such that $p_{\nu i_1} \neq 0$.

To ensure canonicity, we should choose a canonical representative l_1 of the coset Nl. We do not include details of how to do this here, but GAP includes a `CanonicalRightCosetElement` function which fulfills the purpose.[4]

We now need to pick some canonical a_1 such that $p_{\nu i_1} \cdot a_1 \cdot p_{\lambda_1 k} \in Nl$. The formula

$$a_1 = p_{\nu i_1}^{-1} \cdot l_1 \cdot p_{\lambda_1 k}^{-1}$$

suffices, and so we have our representative (i_1, a_1, λ_1).

Chapter 4

Conversion

It may certainly be useful to convert between our linked triple representation of congruences, and other representations. In this section we present a method for finding the linked triple of a congruence, and a method of finding generating pairs from a linked triple. In this way we can move to and from our representation as necessary.

4.1 Generating Pairs to Linked Triple

We start with a function which takes a semigroup congruence (possibly stored as a set of generating pairs) and returns a congruence which is identical mathematically, but is stored as a linked triple. [1, p.84] gives us a simple method for calculating the linked triple of a congruence:

For a congruence ρ on a finite 0-simple Rees 0-matrix semigroup $S = \mathcal{M}^0[G; I, \Lambda; P]$, let $\mathcal{S}$ be the relation on the columns I such that $(i, j) \in \mathcal{S}$ if and only if

$$(i, p_{\lambda i}^{-1}, \lambda)\ \rho\ (j, p_{\lambda j}^{-1}, \lambda)$$

for every row $\lambda \in \Lambda$ such that $p_{\lambda i} \neq 0$ (and therefore $p_{\lambda j} \neq 0$).

Similarly, let $\mathcal{T}$ be the relation on the columns Λ such that $(\lambda, \mu) \in \mathcal{T}$ if and only if

$$(i, p_{\lambda i}^{-1}, \lambda)\ \rho\ (i, p_{\mu i}^{-1}, \mu)$$

for every column $i \in I$ such that $p_{\lambda i} \neq 0$ (and therefore $p_{\mu i} \neq 0$).

Next, choose some $k \in I$ and $\nu \in \Lambda$ such that $p_{\nu k} \neq 0$. Let

$$N = \{a \in G \mid (k, a, \nu)\ \rho\ (k, 1, \nu)\},$$

and we have $(N, \mathcal{S}, \mathcal{T})$, the linked triple of ρ.

Note that, since no row of P is entirely zero, we may always choose $\nu = 1$, and there will be some $k \in I$ such that $p_{\nu k} \neq 0$.

`AsRZMSCongruenceByLinkedTriple` is a GAP implementation of this function, which simply carries out the operation above, and then uses the resulting triple to create a congruence object.

4.2 Linked Triple to Generating Pairs

This function, which we call GeneratingPairsOfCongruence, takes a congruence ρ by linked triple, and returns a list of pairs of elements which can be used to generate the congruence. This list of pairs cannot be relied upon to be minimal, and may contain many more pairs than are necessary to generate the congruence, but it allows for a congruence ρ to be expressed in a different form, for applications which are not well-adapted to the use of linked triples.

The theory is much the same as in Section 4.1, but applied in reverse:

Given the linked triple $(N, \mathcal{S}, \mathcal{T})$, there are pairs which are a consequence of N, pairs which are a consequence of $\mathcal{S}$, and pairs which are a consequence of $\mathcal{T}$. We list all of these in turn.

As in the previous section, we choose arbitrary $k \in I$ and $\nu \in \Lambda$ such that $p_{\nu k} \neq 0$ (again we may always choose $\nu = 1$). Now for every $n \in N$, we add the pair

$$\big((k, n, \nu), (k, 1, \nu)\big)$$

to our list of generating pairs.

Next we consider the column relation $\mathcal{S}$. We have that $(i, p_{\lambda i}^{-1}, \lambda)\ \rho\ (j, p_{\lambda j}^{-1}, \lambda)$ for each $(i, j) \in \mathcal{S}$. To generate all these, for each class $i/\mathcal{S}$ we fix some $i_1 \in i/\mathcal{S}$ and then for each other $j \in i/\mathcal{S}$ add the pair

$$\big((i_1, p_{\lambda i_1}^{-1}, \lambda), (j, p_{\lambda j}^{-1}, \lambda)\big)$$

for each $\lambda \in \Lambda$ such that $p_{\lambda i_1} \neq 0$ (and hence $p_{\lambda j} \neq 0$). A similar method is applied to the row relation $\mathcal{T}$, and we have a full set of generating pairs.

This algorithm is shown in the following pseudo-code:

```
procedure GeneratingPairsOfCongruence($\rho$)
    $R := \varnothing$
    Let $(N, \mathcal{S}, \mathcal{T})$ be the linked triple of $\rho$

    Choose $k \in I,\ \nu \in \Lambda$ such that $p_{\nu k} \neq 0$
    for $n \in N$ do
        Add $\big((k, n, \nu), (k, 1, \nu)\big)$ to R
    end for

    for all equivalence classes $i/\mathcal{S}$ of $\mathcal{S}$ do
        Fix $i_1 \in i/\mathcal{S}$
        for $j \in (i/\mathcal{S}) \setminus \{i_1\}$ do
            for $\lambda \in \Lambda$ such that $p_{\lambda i_1} \neq 0$ do            ▷ and hence $p_{\lambda j} \neq 0$
                Add $\big((i_1, p_{\lambda i_1}^{-1}, \lambda), (j, p_{\lambda j}^{-1}, \lambda)\big)$ to R
            end for
        end for
    end for

    for all equivalence classes $\lambda/\mathcal{T}$ of $\mathcal{T}$ do
        Fix $\lambda_1 \in \lambda/\mathcal{T}$
        for $\mu \in (\lambda/\mathcal{T}) \setminus \{\lambda_1\}$ do
            for $i \in I$ such that $p_{\lambda_1 i} \neq 0$ do            ▷ and hence $p_{\mu i} \neq 0$
                Add $\big((i, p_{\lambda_1 i}^{-1}, \lambda_1), (i, p_{\mu i}^{-1}, \mu)\big)$ to R
            end for
        end for
    end for

    return R
end procedure
```

Chapter 5

Congruence-Free Semigroups

The theory which we have used so far largely applies only to finite 0-simple semigroups. Luckily however, we can extend this theory to find a quick algorithm for determining whether a given semigroup S is congruence-free; this algorithm works for any finite semigroup.

Definition 5.1. A semigroup is **congruence-free** if it has no congruences other than the *universal congruence* $S \times S$ and the *trivial congruence* $\Delta_S = \{(x, x) \mid x \in S\}$. [1, p.93]

First we consider the special case where $|S| \leq 2$.

Theorem 5.2. *Any semigroup S with order 1 or 2 is congruence-free.*

Proof. If $|S| = 1$ then the only congruence is $\{(1, 1)\} = \Delta_S = S \times S$, so S is congruence-free.

If $|S| = 2$, let us define a congruence ρ on S. Let $\{x, y\}$ be the elements of S. By reflexivity, $(x, x), (y, y) \in \rho$. If we add an arbitrary element (x, y) to ρ then by symmetry we must also add (y, x), and we have $\rho = S \times S$. Hence Δ_S and $S \times S$ are the only congruences in S. □

Lemma 5.3. *Any finite congruence-free semigroup is completely simple or completely 0-simple.*

Proof. Let S be a semigroup which is not simple or 0-simple, so that S has a proper ideal I. We can now define a relation ρ_I by

$$\rho_I = (I \times I) \cup \Delta_S.$$

We can show that ρ_I is a congruence as follows:

(R) For each $x \in S$, the pair $(x, x) \in \rho_I$, so ρ_I is reflexive.

(S) For a pair $(x, y) \in \rho_I$, either $x = y$ or $x, y \in I$. In either case, $(y, x) \in \rho_I$ and so ρ_I is symmetric.

(T) Let $(x, y), (y, z) \in \rho_I$. Either $x = y = z$ (in which case $x = z$), or $x, y \in I$ (in which case $z \in I$). In either case $(x, z) \in \rho_I$, so ρ_I is transitive.

(C) Let $(x, y) \in \rho_I$ and $a \in S$. It may be that $x = y$, in which case $ax = ay$ and $xa = ya$. Otherwise $x, y \in I$, and so $ax, ay, xa, ya \in I$ since I is an ideal. In either case, we have $(ax, ay), (xa, ya) \in \rho_I$.

So ρ_I is a congruence. Since I is a *proper* ideal, ρ_I is not equal to Δ_S or $S \times S$, and so S is not congruence-free.

Hence every congruence-free semigroup is simple or 0-simple. Observe that any finite (0-)simple semigroup is completely (0-)simple, and the statement follows. □

We divide all finite semigroups S of order greater than 2 into two cases: either S has a zero, or S has no zero.

Theorem 5.4. *Let S be a finite semigroup without zero, and $|S| > 2$. S is congruence-free if and only if S is a simple group.*

Proof. ($\Rightarrow$): Let S be a finite congruence-free semigroup without zero, with $|S| > 2$. By Lemma 5.3, we know that S is a completely simple semigroup: hence S is isomorphic to some Rees matrix semigroup (without zero) $\mathcal{M}[G; I, \Lambda; P]$. We have two congruences: Δ_S with linked triple $(1, \Delta_I, \Delta_\Lambda)$; and $S \times S$ with linked triple $(G, I \times I, \Lambda \times \Lambda)$.

Assume that N is a proper normal subgroup of G. Hence we have the triple $(N, \Delta_I, \Delta_\Lambda)$: since $(i, j) \in \Delta_I$ or $(\lambda, \mu) \in \Delta_\Lambda$ only if $i = j$ or $\lambda = \mu$ respectively, and $q_{\lambda\lambda ij} = q_{\lambda\mu ii} = 1$, this triple satisfies Definition 1.11, and it is linked. Thus we have another congruence on S, a contradiction. Hence G has no proper normal subgroups, so it must be a simple group, or the trivial group.

If G is trivial, then $|S| = |I| \times |\Lambda|$. Since $|S| > 2$, either $|I| = |\Lambda| = 2$, or at least one of I and Λ has more than 2 elements. If $|I| = |\Lambda| = 2$ we have the linked triples $(1, \Delta_I, \Lambda \times \Lambda)$ and $(1, I \times I, \Delta_\Lambda)$ which represent non-trivial congruences, a contradiction. Alternatively, if one of I and Λ has size greater than 2, then there exists an equivalence $\mathcal{S}$ or $\mathcal{T}$ such that $\Delta_I \subset \mathcal{S} \subset I \times I$ or $\Delta_\Lambda \subset \mathcal{T} \subset \Lambda \times \Lambda$, and so there is a non-trivial congruence given by $(1, \mathcal{S}, \Lambda \times \Lambda)$ or $(1, I \times I, \mathcal{T})$, another contradiction.

Hence G is a simple group. Now, if $|I|$ or $|\Lambda|$ is greater than 1, then we have the linked triple $(G, \Delta_I, \Delta_\Lambda)$ which is not equal to $(1, \Delta_I, \Delta_\Lambda)$ or $(G, I \times I, \Lambda \times \Lambda)$, and we have another congruence. Hence P must be a 1×1 matrix, and so $S \cong G$, so S is a simple group. [1, p.94]

($\Leftarrow$): Conversely, let S be a finite simple group with $|S| > 2$. Clearly S is isomorphic to the Rees matrix semigroup $\mathcal{M}[S; I, \Lambda; P]$ where $|I| = |\Lambda| = 1$ and $p_{11} = 1_S$. Since S is a simple group, there are no normal subgroups except 1 and S. Hence the only linked triples for S are $(1, \Delta_I, \Delta_\Lambda)$ and $(S, \Delta_I, \Delta_\Lambda)$. Hence S has only two congruences and is congruence-free. □

Theorem 5.5. *If S is a finite semigroup with zero, then S is congruence-free if and only if S is isomorphic to a Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$ where G is the trivial group and P has no two identical rows and no two identical columns.*

Proof. ($\Rightarrow$): Let S be a finite congruence-free semigroup with zero. By Lemma 5.3, S must be completely 0-simple, and so by Theorem 1.10 it is isomorphic to some $\mathcal{M}^0[G; I, \Lambda; P]$ with group G and regular P.

The trivial congruence Δ_S has linked triple $(1, \Delta_I, \Delta_\Lambda)$. Consider the triple $(G, \Delta_I, \Delta_\Lambda)$: since $(i, j) \in \Delta_I$ or $(\lambda, \mu) \in \Delta_\Lambda$ only if $i = j$ or $\lambda = \mu$ respectively, and $q_{\lambda\lambda ij} = q_{\lambda\mu ii} = 1$, the triple satisfies Definition 1.11, and it is linked. Hence for S to be congruence-free, we must have $(1, \Delta_I, \Delta_\Lambda) = (G, \Delta_I, \Delta_\Lambda)$, and so $G = 1$.

Now, each matrix entry $p_{\lambda i}$ is equal to 1 or 0, and each relevant $q_{\lambda\mu ij} = 1$. Hence the triple $(1, \varepsilon_I, \varepsilon_\Lambda)$ is linked, and so to be congruence-free we must have $(1, \Delta_I, \Delta_\Lambda) = (1, \varepsilon_I, \varepsilon_\Lambda)$, and so $\varepsilon_I = \Delta_I$ and $\varepsilon_\Lambda = \Delta_\Lambda$. This means that no two rows or columns have zeros in the same places, which, over the range $\{0, 1\}$, is identical to saying that no two rows or columns are equal.

($\Leftarrow$): Let S be a finite Rees 0-matrix semigroup $\mathcal{M}^0[G; I, \Lambda; P]$ with G trivial and P having no two identical rows or columns. Since P is only over the elements $\{0, 1\}$, no two rows or columns have zeros in the same place, and so $\varepsilon_I = \Delta_I$ and $\varepsilon_\Lambda = \Delta_\Lambda$. Hence the only linked triple is $(1, \varepsilon_I, \varepsilon_\Lambda)$, and so S is congruence-free. [1, p.93-94] □

Now that we have considered all finite semigroups, we have an algorithm which decides whether a finite semigroup is congruence-free:

```
Require: $S$ is finite
  procedure IsCongruenceFree($S$)
      if $|S| \leq 2$ then
          return true
      end if
      if $S$ has a zero then
          if $S$ is 0-simple then
              Find $\mathcal{M}^0[G; I, \Lambda; P] \cong S$
              for $i \in \{1 \dots |I| - 1\}$ do
                  for $j \in \{i + 1 \dots |I|\}$ do
                      if Row $i$ = Row $j$ of $P$ then
                          return false
```

```
                    end if
                end for
            end for
            return true
        else
            return false
        end if
    else
        if $S$ is a simple group then
            return true
        else
            return false
        end if
    end if
end procedure
```

Chapter 6

Evaluation

In this chapter, we will show the practical value of these functions, and briefly discuss future directions in which this work could be continued.

6.1 Benchmarking

It was stated in the motivation of this project (see Section 1.1) that computing with congruences using linked triples is computationally easier, and therefore much faster, than by using the representation that currently exists in GAP. We now present two experiments in the GAP system in which a property of a semigroup congruence is calculated – first using the generating pairs representation, and then using the new linked triples representation – and the computation times are compared.

We began by constructing a Rees 0-matrix semigroup R from the principal factor of a random $\mathcal{D}$-class of the full transformation semigroup T_5:

```
gap> S := FullTransformationSemigroup(5);;
gap> R := PrincipalFactor(Random(DClasses(S)));
<Rees 0-matrix semigroup 25x10 over Group([ (1,4,2), (1,2) ])>
```

We then used `CongruencesOfSemigroup` (see Section 2.2) to find a list of all the congruences of R. The congruences are displayed along with a representation of their linked triple (except the universal congruence, a special case):

```
gap> congs := CongruencesOfSemigroup(R);
[ <universal semigroup congruence>,
  <RZMS congruence by linked triple (1,25,10)>,
  <RZMS congruence by linked triple (3,25,10)>,
  <RZMS congruence by linked triple (S3,25,10)> ]
```

We then focused on the third congruence in the list: the congruence with linked triple displayed as `(3,25,10)` – that is, $N = C_3$, $\mathcal{S}$ has 25 equivalence classes, and $\mathcal{T}$ has 10 equivalence classes. We called this congruence `cong`, and then we found its generating pairs representation (see Section 4.2) and called it `pcong`:

```
gap> cong := congs[3];
<RZMS congruence by linked triple (3,25,10)>
gap> pcong := AsSemigroupCongruenceByGeneratingPairs(cong);
<semigroup congruence with 3 generating pairs>
```

Now we have two objects, `cong` and `pcong`, which represent the same congruence. Our first test was to find their equivalence classes using `EquivalenceClasses(cong)` and `EquivalenceClasses(pcong)`. This test was carried out three times, and the times measured; the generating pairs method returned a result in around 52 minutes each time (51m40s, 52m16s and 52m24s respectively), whereas the linked triples method consistently returned a result in less than one tenth of a second.

Next the method for finding the elements of a congruence class was considered (see Section 2.6). Continuing from the previous setup, we defined a Rees 0-matrix semigroup element $x := (24, (1, 4), 4) \in R$, and then we found the elements of its congruence class by calling `ImagesElm(cong, x)` and `ImagesElm(pcong, x)`. Again the test was carried out three times; in each case the generating pairs representation took around 55 minutes to return a result (54m26s, 55m35s and 57m34s respectively), while again the linked triples method returned a result each time in less than one tenth of a second.

6.2 Extensions

As discussed at the end of Chapter 1, the congruences on finite *simple* semigroups can be represented in almost precisely the same way as finite *0-simple* semigroups, but with the removal of complications related to the zero element.[1, p.90-91] To implement and document this would be a natural continuation of this project.

A more substantial extension would be to look into the possibility of extending this theory to represent the congruences of arbitrary semigroups. Furthermore, more consideration could be given to how these congruences describe quotient semigroups and homomorphic images: by the First Isomorphism Theorem (Theorem 1.5.2 in [1, p.23]) congruences and homomorphisms are in a way "equivalent", and it would be good to have implementations of methods which use this link to provide quick calculations on a wider range of objects.

Bibliography

[1] Howie, J.M., *Fundamentals of Semigroup Theory*, Oxford Science Publications, 1995, 3.5-3.7, 83-95.

[2] Holt, D.F., Eick, B., O'Brien, E.A, *Handbook of Computational Group Theory*, Chapman & Hall/CRC Press, 2005.

[3] Ruškuc, N., Lecture Notes, *MT5823 Semigroups*, University of St Andrews, 2014.

[4] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.7.4*; 2014, (`http://www.gap-system.org`).

[5] Mitchell, J.D., *Semigroups - GAP package, Version 2.0*, April 2014.

[6] Hulpke, A., *Computing Normal Subgroups*, Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (Rostock), ACM, New York, 1998, 194198 (electronic)