Enumerating Rees 0-matrix semigroups

Chris Russell

University of St Andrews

cr66@st-andrews.ac.uk

February 26, 2021

Overview

Preliminaries

0-simple semigroups

Definition

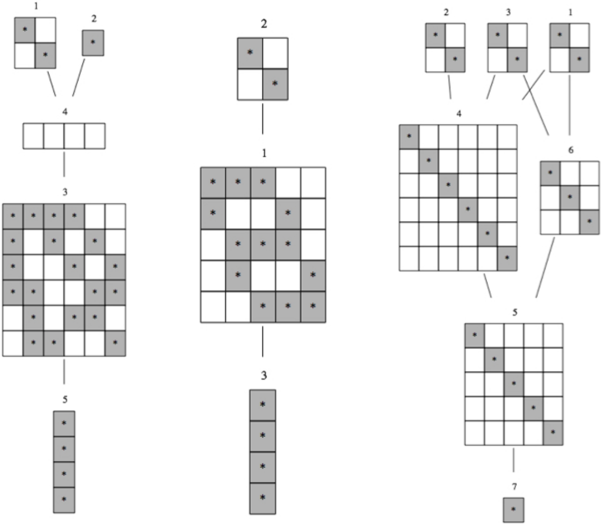
A semigroup S is *0-simple* if its only proper two-sided ideal is $\{0\}$.

The 0-simple semigroups are the key ingredients in a decomposition of semigroups. Intuitively, the Greens $\mathcal{D}$-classes of a semigroup behave like 0-simple semigroups.

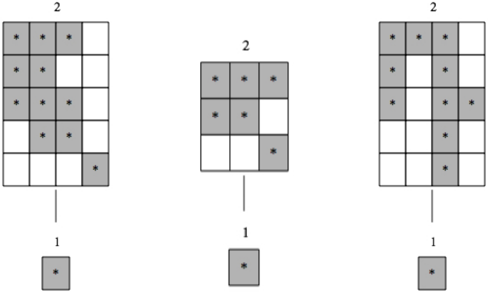
Reminder

Greens $\mathcal{D}$-relation describes the structure of semigroups by relating elements which generate the same (two-sided) ideals.

Decomposition into 0-simple semigroups



Decomposition into 0-simple semigroups



Rees 0-matrix semigroups

Finite Rees 0-matrix semigroups are important due to their correspondence with finite 0-simple semigroups.

Definition

Let G be a group, $P = (p_{i,j})$ be a $m \times n$ matrix with entries in $G \cup \{0\}$. Let $S = (\{1 \cdots m\} \times G \times \{1 \cdots n\}) \cup \{0\}$, and define a multiplication on S by

$$(i, a, j)(k, b, l) = \begin{cases} (i, ap_{j,k}b, l) & \text{if } p_{j,k} \neq 0 \\ 0 & \text{if } p_{j,k} = 0 \end{cases}$$

$$(i, a, j)0 = 0(i, a, j) = 0.$$

Then S is called a *Rees 0-matrix semigroup* denoted $\mathcal{M}^0[G; P]$. If P is 'regular' (every row and column has a non-zero entry) then S is finite 0-simple. Conversely, every finite 0-simple semigroup is isomorphic an RZMS of this kind.

Enumerating RZMS

My work: a database of 0-simple semigroups

I have enumerated the 0-simple semigroups of order less than 50 and plan to release GAP code that anyone can use to generate these. Why? Databases are useful for:

1. Checking hypotheses - users can check whether a result holds for all entries in a database.
2. Obtaining examples - users can filter database entries by certain properties.

Other GAP databases include Small Groups (order < 2000 but not 1024), Primitive Groups (degree less than 4096), Transitive Groups (up to degree 30) and the SmallSemi package (semigroups of order ≤ 8).

What it boils down to (roughly)

Finding all 0-simple semigroups of order k up to isomorphism is equivalent to finding all RZMS (over regular matrices) of order k up to isomorphism.

Proposition

The order of the RZMS $\mathcal{M}^0[G; P]$ equals $|G| * |m| * |n| + 1$ where m, n are the number of rows and columns of P.

We call (G, m, n) the parameters of an RZMS and denote the set of all $m \times n$ matrices over G^0 by $M_{m\times n}(G^0)$. The next lemma lets us consider these cases separately.

Lemma

If the RZMS $\mathcal{M}^0[G; P]$ and $\mathcal{M}^0[H; Q]$ are isomorphic then $G \cong H$ and P, Q have the same dimensions.

The union of RZMS over these matrices in $M_{m\times n}(G^0)$ for all triples (G, m, n) such that $|G| * |m| * |n| + 1 = k$ gives all the RZMS of order k.

A naive approach

As an easy way to start out, I tried the following:

1. **RZMS** $:=$ All Rees 0-matrix semigroups of order k
2. for $\mathbf{i}$ in $\{1 \cdots |\mathbf{RZMS}|\}$ do
3. for $\mathbf{j}$ in $\{1 \cdots \mathbf{i} - 1\}$ do
4. if **RZMS**[**i**] is isomorphic to **RZMS**[**j**] then
5. remove **RZMS**[**j**] from **RZMS**
6. return **RZMS**

Slightly better approach

This approach avoids making as many isomorphism checks.

for (G, m, n) corresponding to order k do
    RZMS $:=$ All Rees 0-matrix semigroups of type (G, m, n)
    for **i** in $\{1 \cdots |\mathbf{RZMS}|\}$ do
        for **j** in $\{1 \cdots \mathbf{i} - 1\}$ do
            if **RZMS[i]** is isomorphic to **RZMS[j]** then
                remove **RZMS[j]** from **RZMS**
    Append **RZMS** to **OUT**
return **OUT**

Speeding things up

Strategy

We have two issues:

1. Isomorphism checks are expensive.
2. There are lots of matrices to check.

Our solutions should be:

1. Reduce number of isomorphism checks and use a better method if possible.
2. Find an easily calculable and significantly smaller subspaces of $M_{m \times n}(G^0)$ which still represent every isomorphism class.

Isomorphism Theorem

Finding an isomorphism is expensive without any specialized method because the general case is hard for semigroups.

RZMS Isomorphism Theorem (simplified)

$\mathcal{M}^0[G; P]$ is isomorphic to $\mathcal{M}^0[G; Q]$ if and only if Q can be obtained from P via:

1. Permuting the rows and columns
2. Multiplying all entries in a row or column by a group element
3. Applying automorphisms of G to every entry at once

We can use this to check whether two RZMS over the same group are isomorphic.

Isomorphism Theorem

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} d & e & f \\ a & b & c \\ g & h & i \end{bmatrix} \qquad \text{(swap rows 1 and 2)}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} a * x & b * x & c * x \\ d & e & f \\ g * y & h * y & i * y \end{bmatrix} \qquad \text{(multiply row 1,3 by x,y)}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} a^{-1} & b^{-1} & c^{-1} \\ d^{-1} & e^{-1} & f^{-1} \\ g^{-1} & h^{-1} & i^{-1} \end{bmatrix}$$

Isomorphism theorem - group actions

The following are all group actions on the space $M_{m\times n}(G^0)$ of $m \times n$ matrices over G^0:

1. Permuting the rows and columns
2. Multiplying all entries in a row or column by a group element
3. Applying automorphisms of G to every entry at once

This is a promising observation - group algorithms are fast and eliminate equivalent cases quickly through symmetries.

The group action

The following group acts faithfully on the space $m \times n$ matrices over G^0.

$$\mathcal{G}(G, m, n) := (Sym(m) \times Sym(n) \times Aut(G)) \rtimes (G^m \times G^n)$$

The action can send P to Q if and only if P and Q create isomorphic RZMS. Thus finding isomorphism classes is equivalent to finding a single representative of every orbit.

1. $Sym(m)$ acts by permuting the m rows and $Sym(n)$ the n columns
2. G^m acts by multiplication on rows and G^n acts by multiplication on columns.
3. $Aut(G)$ acts by applying to every entry at once.

The semidirect product to accounts for these actions not commuting, in general.

Isomorphism Theorem

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} d & e & f \\ a & b & c \\ g & h & i \end{bmatrix} \qquad \text{(swap rows 1 and 2)}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} a * x & b * x & c * x \\ d & e & f \\ g & h & i \end{bmatrix} \qquad \text{(multiply row 1 by x)}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \mapsto \begin{bmatrix} a^{-1} & b^{-1} & c^{-1} \\ d^{-1} & e^{-1} & f^{-1} \\ g^{-1} & h^{-1} & i^{-1} \end{bmatrix}$$

CanonicalImage

The function **CanonicalImage** [Jefferson et al, 2017] takes a group action as well as an element of the set being acted upon and returns a *canonical* element of its orbit. It does this relatively quickly, without needing to find every element of the orbit.

This allows us to stop doing pairwise isomorphism checks. Now we do:

1. **RZMS** := All Rees 0-matrix semigroups of order k
2. for **i** in $\{1 \cdots |\textbf{RZMS}|\}$ do
3.   if **CanonicalImage**(**RZMS**) not in OUT then
4.   add **CanonicalImage**(**RZMS**) to OUT
5. return **OUT**

Checking less matrices

The number of matrices in $M_{m\times n}(G^0)$ is $(|G|+1)^{m*n}$ which scales very quickly. There are a few immediate and easy ways to reduce the workload:

1. The case where m or n equals 1 has only a single isomorphism class.
2. The case where m or n equals 2 is fairly easy to solve and contains relatively few isomorphism classes.
3. The $M_{m\times n}(G^0)$ case corresponds exactly to transposing the $M_{n\times m}(G^0)$ case.

Checking less matrices - linked triples

The theorey of *linked triples* describes all quotients of RZMS. Very breifly, if $\mathcal{M}^0[G; P]$ is an RZMS then for each normal subgroup N of G there exists a homomorphism:

$$\mathcal{M}^0[G; P] \to \mathcal{M}^0[G/N; P']$$

where P' corresponds to P with its entries mapped naturally into the quotient group $p_{i,j} \mapsto p_{i,j}N$. The lets us use the solution to the case $M_{m\times n}((G/N)^0)$ to get a head start in solving the case $M_{m\times n}(G^0)$. Actually, we will just use the $M_{m\times n}(1^0)$ case when solving any case of the form $M_{m\times n}(G^0)$.

Binary matrices

Actually, we will just use the $M_{m\times n}(1^0)$ case when solving any case of the form $M_{m\times n}(G^0)$. Assume these matrices are over some 0-group G^0:

$$\begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \cong \begin{bmatrix} 0 & c \\ a & b \end{bmatrix}$$

and they form isomorphic RZMS. If we replace all non-zero entries by 1 then we will have two matrices which are isomorphic over the trivial 0-group $1^0 = \{0, 1\}$. (The reverse implication does not hold.)

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cong \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Binary matrices

Summary

If P and Q define isomorphic RZMS over some group G and we replace all their entries with 1 then these new matrices will define isomorphic RZMS over the trivial group.

We can then split the case $M_{m\times n}(G^0)$ into separate cases corresponding to the isomorphism classes of $M_{m\times n}(1^0)$.
In the case of a matrix with one zero, we reduce our work by a factor of $m * n$. However, this is of no help for cases where G is trivial.

Binary matrices - performance

This table, roughly, shows the proportion of the work we are cutting out assuming we already have already solved the binary case.

Dimensions	**# binary matrices**	**# regular, up to row&col perms**
3×3	512	14
4×3	4096	37
4×4	65536	115
5×4	1048576	525

And these improvements were quite relevant to cases such as $(G, m, n) = (C_5, 3, 3), (C_4, 4, 3), (C_3, 4, 4), (C_2, 5, 4)$.

Stuck

Solving the $M_{m\times n}(1^0)$ case seems to be a hard problem. It may be rephrased as finding binary matrices up to the equivalence of row and column permutations but that doesn't help. My search ended after finding the 14289957 matrices in the 8×6 case and with me being unable to compute the 7×7 case.

Other challenges

1. Normalized RZMS.
2. Finding binary matrices up to row and column permutation equivalence.
3. Turning the group action into a permutation group action (the best GAP algorithms are for permutation groups acting on positive integers).
4. Storing solved cases and reaccessing them when needed.
5. Parallelising computations.

References

J. M. Howie (1995)
Fundamentals of Semigroup Theory

Christopher Jefferson, Eliza Jonauskyte, Markus Pfeiffer, Rebecca Waldecker (2017)
Minimal and Canonical Images