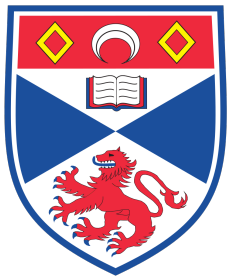

MT5991 Professional Skills for Mathematical Scientists

The Order Problem for Natural and Tropical Matrix Semigroups in GAP

Author:
Stuart Burrell

Supervisor:
Dr James Mitchell

School of Mathematics and Statistics

May, 2016

Abstract

We survey the literature required to address the order problem for natural and tropical matrix semigroups. In addition, we implement a variety of methods in GAP, culminating in an effective decision algorithm for the order problem of natural and tropical matrix semigroups. The intended readership is the graduate student in pure mathematics who is unfamiliar with this area. Thus, although we may recap crucial definitions for clarity, a strong mathematical knowledge base is expected.

Contents

Chapter 0

Introduction

Numerous methods in the GAP semigroups package [1] [2] benefit from an efficient and terminating procedure for detecting semigroups of infinite cardinality. In addition, this may aid research when experimenting with novel and unknown semigroups. Determining the existence of an algorithm that decides the cardinality of an object is known as the order problem. To begin, we focus on semigroups of matrices over $\mathbb{N} = \{0, 1, 2, ...\}$, which we refer to as natural, or positive integer, matrices. A positive solution to the order problem for natural matrix semigroups is given in [3]. The objective of chapter 1 is to present an implementation of a decision algorithm for the order problem of natural matrix semigroups in GAP. Preceding this, we give a survey of the theoretical results from [3] that justify our approach.

The second objective of our work is to consider the order problem for semigroups of tropical matrices. There is an ambiguity in current literature concerning their definition. We use the max-plus convention and adopt the terminology *max-plus* matrices. That is, we consider matrices over the max-plus semiring. Denoted $\mathbb{R}_{\max}$, this is the semiring consisting of the set $\mathbb{R} \cup \{-\infty\}$ with the operations defined by $a \oplus b = \max\{a, b\}$ and $a \otimes b = a + b$. A solution to the order problem for finitely generated semigroups of max-plus matrices is given in [4], though the required theory is more involved. The approach draws on spectral theory, which is the analogy of Perron-Frobenius theory in this context, and a graphical interpretation of

max-plus matrices.

In chapter 2, we present and justify implementations of several required methods for max-plus matrices. These include an inversion algorithm from [5], an elementary procedure for constructing a directed graph from a max-plus matrix, and, finally, a related method concerning matrix irreducibility. Subsequently, chapter 3 deals with finding the circuits of a graph that do not contain subcircuits, known as simple circuits. We discuss and present an implementation of Johnson's algorithm [6] in GAP. Though seemingly tangential, several of our later algorithms depend upon it.

Chapter 4 first focuses on computing the largest eigenvalue of a max-plus matrix, known as the spectral radius. Thematically, we use a graphical approach based on a classical result discussed in [4] and [7], amongst others. This is fundamental. In most cases a simple computation based on the spectral radius can allow rapid detection of infinite cardinality. Next, we consider the eigenvector corresponding to the spectral radius, and a process of semigroup normalisation crucial to the solution to the order problem in [4].

We proceed to the climax of our work in Chapter 5. Drawing on the methods developed throughout, we present an implementation in GAP to decide the order problem for max-plus matrix semigroups. In order to elaborate on our approach, recall that an element a of a semigroup S is torsion if and only if there exists $m, n \in \mathbb{N}$ such that $m > n \geq 1$ and $a^m = a^n$. We then say a semigroup is torsion if each of its elements is torsion. In addition, the famous Burnside problem asks whether a finitely generated torsion semigroup is finite. Gaubert gives a positive solution to the Burnside problem for max-plus matrix semigroups in [4]. This reduces the order problem to the analogously defined torsion problem for max-plus matrix semigroups. Consequently, our implementation to decide the order problem is based on a decider for the torsion problem from [4]. Finally, in chapter 6, we summarise our work

and verify the effectiveness of our implementation with a series of tests.

Chapter 1

The Order Problem for Natural Matrix Semigroups

The purpose of this section is to present an implementation in GAP that decides the order problem for finitely generated natural matrix semigroups. Prior to this, we present the theoretical justification of this implementation from [3], where the algorithm was first introduced.

First, for a semiring R, let $M_n(R)$ denote the set of $n \times n$ matrices over R. Then, let $S = \langle M_1, M_2, ..., M_k \rangle$ denote the semigroup generated by $\{M_1, ..., M_k\}$, where $M_i \in M_n(\mathbb{N})$ for $i = 1, ..., k$. We wish to determine in finite time if $|S| < \infty$.

Next, consider the semiring $\mathbb{N}_2 = \{0, 1, 2\} \subseteq \mathbb{N}$ with the operations $a \oplus b = \min\{a + b, 2\}$ and $a \otimes b = \min\{ab, 2\}$. The following method is based on mapping elements from $M_n(\mathbb{N})$ into the finite set $M_n(\mathbb{N}_2)$. In particular, we define $\Psi : M_n(\mathbb{N}) \to M_n(\mathbb{N}_2)$ by

$$(A\Psi)_{ij} = \min\{A_{ij}, 2\}. \tag{1.1}$$

In order to prove the first key theorem, we require the following two lemmas from [3], recalling the definition of a torsion element from chapter 0.

Lemma 1.0.1. *Let A and B be matrices in $M_n(\mathbb{N})$, such that $A\Psi = B\Psi$. Then A is torsion if and only if B is torsion.*

Proof. First, assume that A is torsion. We wish to show that B is torsion. Let C be a matrix such that $A\Psi = C\Psi$ and $C_{ij} = A_{ij}$ for all but a single pair $(i, j) = (a, b)$. It is sufficient to show that C is torsion, since our argument can be iterated by the transitivity to obtain the result for B. Recall $A\Psi = B\Psi$ by assumption, hence $(C\Psi)_{ab} = (A\Psi)_{ab}$. Thus, from the definition of Ψ and since $C_{ab} \neq A_{ab}$, it follows that $C_{ab}, A_{ab} \geq 2$.

To proceed, let $\mathbb{N}(x)$ denotes the set of polynomials with natural number coefficients. We consider an indeterminate variable x, and construct a matrix $A' \in M_n(\mathbb{N}(x))$ such that $A'_{ij} = A_{ij}$ for $(i, j) \neq (a, b)$ and $A'_{ab} = x$. For a fixed value x_0, we write $A'(x_0)$ to be the evaluation of A' at x_0. Next, consider the family of polynomials given by

$$P = \{(A')^m_{ij} \,|\, m \in \mathbb{N},\, i, j = 1, ..., n\}. \tag{1.2}$$

Suppose that P has infinite cardinality. Then, since each element $(A')^m_{ij}$ of P is a polynomial of x with natural number coefficients, the set $\{(A')^m_{ij}(A_{ab}) \,|\, m \in \mathbb{N},\, i, j = 1, ..., n\}$ is unbounded since $A_{ab} \geq 2 > 0$. This is a contradiction, since A is torsion if and only if there exists a $k \geq 0$ such that $A^m_{ij} \leq k$ for all i, j and m. Hence P is finite and so A' is torsion, since only finitely many polynomial expressions occur as valid entries on taking powers. In particular, it follows that $C = A'(C_{ab})$ is torsion, as required. □

For the next lemma, recall that $a \in S$ is idempotent if and only if $a^2 = a$.

Lemma 1.0.2. *Let P be a torsion semigroup and let $\gamma : S \to P$ be a morphism, such that for all $a \in S$, if $a\gamma$ is idempotent then a is torsion. Then S is torsion.*

Proof. Suppose $a \in S$. By assumption P is torsion, and so $a\gamma$ is torsion. By definition, the there exist n, m such that $(a\gamma)^n = (a\gamma)^m$. Hence, $(a\gamma)^k = (a\gamma)^{k+(m-n)}$ for $k \geq n$ by

periodicity. Let k be such that $k = q(m-n)$ for some $q \in \mathbb{N}$. It follows that

$$((a\gamma)^k)^2 = (a\gamma)^{2k} = (a\gamma)^{k+k} = (a\gamma)^{k+q(m-n)} = (a\gamma)^k, \tag{1.3}$$

and so $(a\gamma)^k$ is idempotent. By the hypothesis, it then follows that a^k is torsion. Hence there exist m, n such that $(a^k)^m = (a^k)^n$. It then immediately follows that $a^{km} = a^{kn}$ and so a is torsion. Since a was arbitrary, S is torsion. □

To proceed, we require the following result by McNaughton and Zalcstein. Recall that a semigroup is locally finite if every finitely generated subsemigroup is finite.

Theorem 1.0.3. *For a field F and $n \in \mathbb{N}$, every torsion semigroup $S \subseteq M_n(F)$ is locally finite.*

We can now prove the following theorem, which is central to the decision algorithm. First, let $\iota : M(\mathbb{N}_2) \to M(\mathbb{N})$ denote the set inclusion map.

Theorem 1.0.4. *A finitely generated subsemigroup $S \subseteq M_n(\mathbb{N})$ is finite if and only if for all $A \in S\Psi$, if A is idempotent then $A\iota$ is torsion.*

Proof. First, we assume that S is finite and thus torsion. Let $A \in S\Psi$ be idempotent. We wish to show that $A\iota$ is torsion. Suppose $B \in S$ is such that $B\Psi = A$. Since $A \in S\Psi$, clearly $A = A\iota = A\iota\Psi$. Hence $B\Psi = A\iota\Psi$. By lemma 1.0.1, this implies B is torson if and only if $A\iota$ is torsion. $B \in S$ is torsion since S is torsion by assumption. Hence $A\iota$ is torsion, as required.

Next, assume that for all $A \in S\Psi$, if A is idempotent then $A\iota$ is torsion. We wish to show that S is finite. Let $A \in S$ be such that $A\Psi$ is idempotent. By hypothesis $A\Psi\iota$ is torsion. Thus, A is torsion by lemma 1.0.1, since $A\Psi = (A\Psi\iota)\Psi$. The conditions of lemma 1.0.2 are met, and thus S is torsion. The result now follows from 1.0.3, since S is finitely generated and we can consider the elements of S as a subset of $M_n(\mathbb{Q})$, noting that $\mathbb{Q}$ is a field. □

Theorem 1.0.4 reduces the order problem to the torsion problem for matrices $A \in S$ such that $A\Psi$ is idempotent. The following two lemmas exploit this reduction, and form an easily verifiable criteria to decide the torsion problem for such matrices. These rely on the notion of a precedence digraph for a matrix $A \in M_n(\mathbb{N})$, denoted $\mathcal{P}(A)$. In particular, $\mathcal{P}(A) = (V, E \times E, L)$ for vertex set $V = \{1, ..., n\}$ and edge set $E \subseteq V \times V$, where $(i, j) \in E$ if and only if $A_{ij} \neq 0$. Furthermore, $L = \{A_{ij} : (i, j) \in E \times E\}$ represents a set of edge labels, or weights. We require the following technical lemma from [3]. For brevity, we omit the proof, but note that it follows from the key fact that for all $k \in \mathbb{N}$, A^k_{ij} is the sum of the labels of the length k directed paths from vertex i to j in $\mathcal{P}(A)$.

Lemma 1.0.5. *Let $A \in M_n(\mathbb{N})$. Then the following statements are equivalent*

(a) *A is torsion*

(b) *$\mathcal{P}(A)$ contains neither a directed circuit with label at least 2, nor two distinct circuits joined by a path.*

(c) *There exists a permutation matrix P such that $P^{-1}AP$ has the block form*

$$\begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & B_{33} \end{pmatrix}, \tag{1.4}$$

where B_{11} and B_{33} are upper-triangular matrices and B_{22} is a permutation matrix.

Lemma 1.0.5 allows us to prove the following corollary, which establishes an important result concerning idempotent elements in the image of S under Ψ.

Corollary 1.0.6. *Suppose $A \in M_n(\mathbb{N})$ is torsion. If $A\Psi$ is idempotent, then $A^2 = A^3$.*

Proof. Since A is torsion, by lemma 1.0.5 there exists a permutation matrix P such that

$P^{-1}AP$ has the form

$$\begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & B_{33} \end{pmatrix}. \tag{1.5}$$

By cancellation, $(P^{-1}AP)^2 = P^{-1}A^2P$. Thus, since $A\Psi$ is idempotent and Ψ is a morphism we have that

$$\begin{aligned} (P^{-1}AP)^2\Psi &= (P^{-1}A^2P)\Psi \\ &= (P^{-1}\Psi)(A^2\Psi)(P\Psi) \\ &= (P^{-1}\Psi)((A\Psi)^2)(P\Psi) \\ &= (P^{-1}\Psi)((A\Psi))(P\Psi) \\ &= (P^{-1}AP)\Psi. \end{aligned} \tag{1.6}$$

Next, by direct computation, observe that

$$(P^{-1}AP)^2 = \begin{pmatrix} (B_{11})^2 & (B_{11}B_{12} + B_{22}B_{12}) & \dots \\ 0 & (B_{22})^2 & (B_{22}B_{23} + B_{23}B_{33}) \\ 0 & 0 & (B_{33})^2 \end{pmatrix}. \tag{1.7}$$

Thus, since $(P^{-1}AP)^2\Psi = (P^{-1}AP)\Psi$, we deduce that $((P^{-1}AP)^2)_{12}$ and $((P^{-1}AP)^2)_{23}$ imply B_{11} and B_{33} are null, since B_{22} is a permutation matrix (and hence not null). Thus, there exists a permutation matrix T such that

$$(T^{-1}AT) = \begin{pmatrix} 0 & C & D \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix}, \tag{1.8}$$

by choosing T to permute the rows such that permutation matrix B_{22} becomes the identity,

I. Then, a direct computation yields

$$(T^{-1}AT)^2 = \begin{pmatrix} 0 & C & D \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & C & D \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & C & EC \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} \tag{1.9}$$

and

$$\begin{aligned}(T^{-1}AT)^3 = \begin{pmatrix} 0 & C & EC \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & C & D \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} &= \begin{pmatrix} 0 & C & EC \\ 0 & I & E \\ 0 & 0 & 0 \end{pmatrix} \\ &= (T^{-1}AT)^2.\end{aligned} \tag{1.10}$$

It then immediately follows that $T^{-1}A^2T = T^{-1}A^3T$. By cancellation, we deduce $A^2 = A^3$, as required. □

Observe that corollary 1.0.6 implies that, for $A \in M_n(\mathbb{N})$, if $A\Psi$ is idempotent and $A^2 \neq A^3$, then A is not torsion. From this, we have the next theorem which immediately suggests a clear algorithmic process for our implementation.

Theorem 1.0.7. *Let S be a finitely generated subsemigroup of $M_n(\mathbb{N})$. Then S is finite if and only if for all $A \in S\Psi$, if A is idempotent then $(A\iota)^2 = (A\iota)^3$.*

Proof. Assume that S is finite, and suppose $A \in S\Psi$ is idempotent. We wish to show that $(A\iota)^2 = A(\iota)^3$. Since $A\iota\Psi = A$, by corollary 1.0.6 it suffices to show that $A\iota$ is torsion. This follows immediately from theorem 1.0.4, as required.

Conversely, assume that for all $A \in S\Psi$, if A is idempotent then $(A\iota)^2 = (A\iota)^3$. We wish to show that S is finite. $A\iota$ is torsion, since $(A\iota)^2 = (A\iota)^3$. Hence by theorem 1.0.6 S is finite, as required. □

Theorem 1.0.7 enables us to decide the order problem for finitely generated natural matrix semigroups is as follows. First, compute the finite semigroup $S\Psi$. Then, verify that each

idempotent element $A \in S\Psi$ satisfies $(A\iota)^2 = (A\iota)^3$. To conclude this section, we present an implementation of this algorithm for inclusion in the GAP semigroups package.

```
InstallMethod(IsFinite,
"for a semigroup of matrices of positive integers",
[IsIntegerMatrixSemigroup],
function(S)
  local gens, ET, mat, row, val;
  gens := GeneratorsOfSemigroup(S);
  for mat in gens do
    for row in mat do
      for val in row do
        if val < 0 then
          TryNextMethod();
        fi;
      od;
    od;
  od;
  ET := Idempotents(Semigroup(List(gens, x -> AsMatrix(IsNTPMatrix, x
   , 1, 2))));
  for mat in ET do
    mat := AsMatrix(IsIntegerMatrix, mat);
    if mat ^ 2 <> mat ^ 3 then
      return false;
    fi;
  od;
  return true;
```

```
24 end);
```

Chapter 2

Tropical Matrices

We begin with an introduction to max-plus matrices and their arithmetic. Next, we draw on [5] to prove and present an effective inversion algorithm. This is utilised in chapter 5. Moving on, we give a GAP method for generating the precedence digraph for max-plus matrices. Finally, we present a method for extracting the blocks, or partitions, of max-plus matrices that correspond to the strongly connected components of the precedence digraph.

2.1 Basic definitions

We denote the set of $n \times n$ max-plus matrices by $\mathbb{R}_{\max}^{n\times n}$. Arithmetic in $\mathbb{R}_{\max}^{n\times n}$ is naturally defined. For $A, B \in \mathbb{R}_{\max}^{n\times n}$, we define the sum $A \oplus B$ by

$$[A \oplus B]_{ij} = \max\{A_{ij}, B_{ij}\}. \tag{2.1}$$

Furthermore, we define the product $A \otimes B$ by

$$[A \otimes B]_{ij} = \bigoplus_{k\in\{1,\ldots,n\}} A_{ik} \otimes B_{kj} = \max_{k\in\{1,\ldots,n\}} \{A_{ik} + B_{kj}\} \tag{2.2}$$

In addition, for a scalar $\gamma \in \mathbb{R}_{\max}$ we define

$$(\gamma A)_{ij} = (\gamma \otimes A)_{ij} = \gamma \otimes A_{ij} = \gamma + A_{ij}, \tag{2.3}$$

for all $i, j = 1, ..., n$. Next, note that the additive ($\oplus$) and multiplicative ($\otimes$) identities of $\mathbb{R}_{\max}$ are $-\infty$ and 0, sometimes denoted $\mathbb{0}$ and $\mathbb{1}$, respectively. Correspondingly, the identity matrix $I \in \mathbb{R}_{\max}^{n\times n}$ is defined by

$$I_{ij} = \begin{cases} \mathbb{1} = 0 & i = j \\ \mathbb{0} = -\infty & i \neq j \end{cases}. \tag{2.4}$$

Relatedly, we say that $D \in \mathbb{R}_{\max}^{n\times n}$ is diagonal if $D_{ij} = -\infty$ for $i \neq j$. Furthermore, we define $Z \in \mathbb{R}_{\max}^{n\times n}$ such that $Z_{ij} = -\infty = \mathbb{0}$ for all i, j to be the zero max-plus matrix. To conclude this section, we present a simple example to clarify arithmetic in $\mathbb{R}_{\max}^{n\times n}$.

Example 2.1.1.

Let $A, B \in \mathbb{R}_{\max}^{n\times n}$ be such that $A = \begin{pmatrix} -\infty & 2 \\ -4 & 0 \end{pmatrix}$ and $B = \begin{pmatrix} 0 & 1 \\ -3 & 4 \end{pmatrix}$. Then

$$A \oplus B = \begin{pmatrix} \max\{-\infty, 0\} & \max\{2, 1\} \\ \max\{-4, -3\} & \max\{0, 4\} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ -3 & 4 \end{pmatrix}, \tag{2.5}$$

and

$$\begin{aligned} A \otimes B &= \begin{pmatrix} \max\{(-\infty + 0), (2 + (-3))\} & \max\{(-\infty + 1), (2 + 4)\} \\ \max\{(-4 + 0), (0 + (-3))\} & \max\{(-4 + 1), (0 + 4)\} \end{pmatrix} \\ &= \begin{pmatrix} -1 & 6 \\ -3 & 4 \end{pmatrix}. \end{aligned} \tag{2.6}$$

2.2 Inversion

Our later work requires a process of inversion for diagonal max-plus matrices. As you shall see, this case is simple. However, we include a general method from [5] to clarify inversion for the diagonal case, and to provide a function of general utility for the GAP semigroups package.

First, for $\boldsymbol{\lambda} = (\lambda_1...., \lambda_n)$, we denote by $D(\boldsymbol{\lambda})$ the $n \times n$ diagonal matrix with $D_{ii} = \lambda_i$ for $i \in \{1, ..., n\}$. In addition, we say P is a max-plus permutation matrix if each row and column in P contains exactly one entry equal to 0 and all other entries equal to $-\infty$. Equivalently, we denote by P_σ the max-plus permutation matrix obtained through a permutation σ of the rows of the identity I. We are now ready to present a result and proof from [5], which characterises the invertible max-plus matrices and indicates the inversion process.

Theorem 2.2.1. *Let $A \in \mathbb{R}_{\max}^{n\times n}$. Then A has an inverse if and only if $A = P_\sigma \otimes D(\boldsymbol{\lambda})$. Moreover, the inverse of A is equal to $P_{\sigma^{-1}} \otimes D(-\boldsymbol{\lambda})$.*

Proof. To begin, we establish the result for right inverses. Suppose there exists a $B \in \mathbb{R}_{\max}^{n\times n}$ such that $A \otimes B = I$. By definition, this implies

$$\max_k\{A_{ik} + B_{kj}\} = \begin{cases} 0 & i = j \\ -\infty & i \neq j \end{cases}. \tag{2.7}$$

Thus, for each i there exists a k such that $A_{ik} + B_{ki} = 0$. It immediately follows that there exists a function $\theta(i)$ such that $A_{i\theta(i)} + B_{\theta(i)i} = 0$. Next, observe that for $i \neq j$ we have

$$A_{i\theta(j)} + B_{\theta(j)j} = -\infty, \tag{2.8}$$

by 2.7. However, since $A_{j\theta(j)} + B_{\theta(j)j} = 0$, this implies that $A_{i\theta(j)} = -\infty$ for $i \neq j$. It follows that θ is injective, since if $\theta(i) = \theta(j)$ for $i \neq j$, then $A_{i\theta(i)} = A_{i\theta(j)} \implies 0 = -\infty$, a contradiction. Hence θ is a permutation on $\{1, ..., n\}$ and each row of A contains a singe

entry not equal to $-\infty$. In addition, since $A_{i\theta(i)} > A_{i\theta(j)}$ for $i \neq j$, the unique entry in the $\theta(i)^{\text{th}}$ column not equal to $-\infty$ is $A_{i\theta(i)}$.

Next, consider the matrix $A' = P_\theta \otimes A$. The permutation matrix P_θ permutes the rows of A such that the i^{th} row of A becomes the $\theta(i)^{\text{th}}$ row of A'. Thus, A' is a diagonal matrix, since θ is a permutation and the i^{th} row of A has a single entry not equal to $-\infty$ in the $\theta(i)^{\text{th}}$ column. It then follows that

$$P_\theta \otimes A = A' = D(\boldsymbol{\lambda}), \tag{2.9}$$

for some $\boldsymbol{\lambda} \in \mathbb{R}^n_{\max}$. On setting $\sigma = \theta^{-1}$, we have

$$P_\theta \otimes A = D(\boldsymbol{\lambda}) \implies A = P_\sigma \otimes D(\boldsymbol{\lambda}), \tag{2.10}$$

since $P_\sigma \otimes P_\theta = I$. For the converse, direct calculation imples $A^{-1} = P_{\sigma^{-1}} \otimes D(-\boldsymbol{\lambda})$.

Finally, we show that a right inverse is also a left inverse. Suppose B is a right inverse of A, then A is of the form $A = P_\sigma \otimes D(\boldsymbol{\lambda})$. Observe $B' = D(-\boldsymbol{\lambda}) \otimes P_{\sigma^{-1}}$ is a left inverse of A. But we have that

$$B' = B' \otimes I = B' \otimes (A \otimes B) = (B' \otimes A) \otimes B = B, \tag{2.11}$$

as required. □

To conclude, we present an implementation of an inversion algorithm in GAP, derived from theorem 2.2.1.

```
InstallMethod(InverseOp, "for a max-plus matrix",
[IsMaxPlusMatrix],
function(mat)
```

```
  local dim, seen_rows, seen_cols, out, row, col;

  dim := DimensionOfMatrixOverSemiring(mat);
  seen_rows := BlistList([1 .. dim], []);
  seen_cols := BlistList([1 .. dim], []);
  out := List([1 .. dim], x -> List([1 .. dim], y -> -infinity));

  for row in [1 .. dim] do
    for col in [1 .. dim] do
      if mat[row][col] <> -infinity then
        if seen_rows[row] or seen_cols[col] then
          return fail;
        fi;
        seen_rows[row] := true;
        seen_cols[col] := true;
        out[col][row] := -mat[row][col];
      fi;
    od;
  od;

  return Matrix(IsMaxPlusMatrix, out);
end);
```

2.3 The precedence digraph

A variety of our methods, particularly the calculation of the Spectral Radius in chapter 4, require the notion of a precedence digraph that we introduced in chapter 1. In this short section we give a definition of $\mathcal{P}(A)$ in the max-plus context and present a GAP method for computing the precedence digraph of a max-plus matrix.

For $A \in \mathbb{R}_{\max}^{n\times n}$, we define $\mathcal{P}(A) = (V, E, L)$ for vertex set $V = \{1, ..., n\}$ and edge set $E \subseteq V \times V$, where $(i, j) \in E \times E$ if and only if $A_{ij} \neq -\infty$. As before, $L = \{A_{ij} : (i, j) \in E \times E\}$ represents a set of edge labels, or weights. Observe there exists a single minor adaptation from the previous definition. In particular, $E \times E$ is defined according to the max-plus additive identity $-\infty$, as opposed 0 as in the natural matrix case.

The following method returns a precedence digraph of a max-plus matrix with the exception that $L = \emptyset$ is fixed. It is typically computationally unnecessary to attach the weights to the digraph object, since they can be readily accessed from the input matrix. We make use of the GAP Digraphs package [8].

```
1  InstallMethod(UnweightedPrecedenceDigraph,
2  "for a max-plus matrix",
3  [IsMaxPlusMatrix],
4  function(mat)
5    local adj;
6
7    # Auxiliary function used to compute the adjacency matrix
8    adj := function(i, j)
9      if mat[i][j] = -infinity then
10       return false;
```

```
    else
      return true;
    fi;
  end;

  # Generate and return digraph object
  return Digraph([1 .. DimensionOfMatrixOverSemiring(mat)], adj);
end);
```

2.4 Irreducible blocks

The idea of matrix irreducibility is central to the positive solution of the torsion problem for max-plus matrix semigroups in [4]. We say that a matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is irreducible if and only if

$$\forall i, j : i \neq j : \exists k \in \mathbb{N} : (A^k)_{ij} \neq -\infty. \tag{2.12}$$

Moreover, recall that $(A^k)_{ij}$ is equal to the sum of the labels of the length k paths from vertex i to j in $\mathcal{P}(A)$ (noting that sum refers to $\oplus$ in this context). Thus, the above admits a translation: $A \in \mathbb{R}_{\max}^{n\times n}$ is irreducible if and only if $\mathcal{P}(A)$ is strongly connected. This motivates the definition of an irreducible block. First, we say a strongly connected component V_c is maximal if there does not exist a vertex $v \notin V_c$ such that $V_c \cup \{v\}$ is strongly connected. Then, for each maximal strongly connected component with vertex set V_c, an irreducible block of A is formed by deleting the rows and columns from A corresponding to vertices $V \setminus V_c$. This definition is essential for deciding the torsion problem, since a case arises that requires the verification of a property for each irreducible block of a matrix. Thus, we present a GAP method that returns a list of the irreducible blocks for a given max-plus matrix.

```
InstallMethod(IrreducibleBlocks,
"for a max–plus matrix",
[IsMaxPlusMatrix],
function(mat)
  local prec, scc;

  # Compute strongly connected components of precedence graph of M
  prec := UnweightedPrecedenceDigraph(mat);
  scc := DigraphStronglyConnectedComponents(prec).comps;

  # Reverse engineer maximal irreducible block matrices
  return List(scc, c -> Matrix(IsMaxPlusMatrix,
                               List([1..Length(c)],
                                    x -> List([1..Length(c)],
                                              y -> mat[c[x]][c[y]]))))
   );
end);
```

Chapter 3

Computing the Simple Circuits of a Directed Graph

Simple circuits of the precedence digraph play an important role in our overall approach. Recall that a circuit is a path of vertices $(v_1, v_2, v_3, ...v_k)$ with $v_1 = v_k$. In addition, we say that a circuit is simple, or elementary, if $v_1, ..., v_{k-1}$ are distinct. A naive computation to find all such simple circuits is typically computationally expensive. However, a variety of more sophisticated algorithms exist to increase efficiency, such as those by Tiernan [9], Tarjan [10], and Johnson [6]. Of these, Johnson's algorithm is best with time complexity $O((v+e)(c+1))$ and space complexity $O(v+e)$, where v, e and c are the number of vertices, edges and circuits in the digraph, respectively. We provide a brief description of Johnson's algorithm and give an implementation for inclusion in the GAP Digraphs package.

3.1 Johnson's algorithm

In Johnson's algorithms we iterate over a set of root vertices, at each stage computing a collection of simple circuits that contain the current root. Suppose $V = \{1, .., n\}$. The basic process of this approach is to take a root $r \in V$ and compute all simple circuits in the sub-

graph induced by $V_r = \{r, r+1, ..., n\}$ that contain r. Recall that an induced subgraph is the graph consisting of vertex set V_r and edge set $E_r = \{(i,j) \in E | i,j \in V_r\}$. Naturally, we begin with $r = 1$ and increment by one.

The process of finding simple circuits in Johnson's algorithm is a sophisticated instance of depth-first search. It uses a recursive structure, based on a function `CIRCUIT`. We build simple circuits one at a time by use of a stack. Calling the function `CIRCUIT` appends a possible vertex to the end of a stack, and deletes it on a return from this call. Due to the recursive structure, this is not trivial, since prior to the return subsequent calls to `CIRCUIT` will be made. This constructs a path in the stack, which will be stored if it is a circuit before the deletion occurs. Strictly speaking, our implementation only mimics a stack, but the effect is equivalent. For details, see the code at the end of the section.

The crucial feature of this algorithm is that circuits are constructed by use of a blocking function. This constitutes the main improvement of Johnson's algorithm over the alternatives [9] and [10]. A blocked vertex cannot be added to the stack by `CIRCUIT`, and a vertex is blocked immediately upon being added to the stack. This mechanism allows us to construct simple circuits in the following way. `CIRCUIT` is recursively applied until no accessible vertices from the root remain unblocked, and then, if the path in the stack is a circuit we append it to the output variable. In short, vertices are unblocked upon the finding a new circuit or returning to the outer procedure and selecting a new root. There is an added level of sophistication to increase efficiency, in that we store information of previously blocked vertices to aid the unblocking process. This retains information from previous unsuccessful searches. Essentially, this allows us to only unblock the minimum amount of required vertices. Consequently, efficiency is increased, since blocked vertices simulate a decrease in instance size.

Next, we present an implementation of Johnson's algorithm in GAP for inclusion in the

Digraphs package. This provides the technical details to support the above description. The author would like to thank W. Wilson for providing some minor alterations and debugging assistance.

```
InstallMethod(DigraphAllSimpleCircuits,
"for a digraph",
[IsDigraph],
function(digraph)
  local UNBLOCK, CIRCUIT, out, stack, endofstack, gr, scc, n, blocked
   , B,
  gr_comp, comp, s, loops, i;

  if DigraphNrVertices(digraph) = 0 or DigraphNrEdges(digraph) = 0
   then
    return [];
  fi;

  UNBLOCK := function(u)
    local w;
    blocked[u] := false;
    while not IsEmpty(B[u]) do
      w := B[u][1];
      Remove(B[u], 1);
      if blocked[w] then
        UNBLOCK(w);
      fi;
    od;
```

```
22  end;
23
24  CIRCUIT := function(v, component)
25    local f, buffer, dummy, w;
26
27    f := false;
28    endofstack := endofstack + 1;
29    stack[endofstack] := v;
30    blocked[v] := true;
31
32    for w in OutNeighboursOfVertex(component, v) do
33      if w = 1 then
34        buffer := stack{[1 .. endofstack]};
35        Add(out, DigraphVertexLabels(component){buffer});
36        f := true;
37      elif blocked[w] = false then
38        dummy := CIRCUIT(w, component);
39        if dummy then
40          f := true;
41        fi;
42      fi;
43    od;
44
45    if f then
46      UNBLOCK(v);
47    else
48      for w in OutNeighboursOfVertex(component, v) do
```

```
      if not w in B[w] then
        Add(B[w], v);
      fi;
    od;
  fi;

  endofstack := endofstack - 1;
  return f;
end;

out := [];
stack := [];
endofstack := 0;

gr := DigraphRemoveLoops(ReducedDigraph(digraph));
if DigraphVertexLabels(digraph) <> DigraphVertices(digraph) then
  SetDigraphVertexLabels(gr, Filtered(DigraphVertices(digraph),
                                      x -> OutDegrees(digraph) <>
 0));
fi;

# Strongly connected components of the reduced graph
scc := DigraphStronglyConnectedComponents(gr);

# B and blocked only need to be as long as the longest connected
 component
n := Maximum(List(scc.comps, Length));
```

```
74  blocked := BlistList([1 .. n], []);
75  B := List([1 .. n], x -> []);
76
77  # Perform algorithm once per connected component of the whole
     digraph
78  for gr_comp in scc.comps do
79    n := Length(gr_comp);
80    if n = 1 then
81      continue;
82    fi;
83    gr_comp := InducedSubdigraph(gr, gr_comp);
84    comp := gr_comp;
85    s := 1;
86    while s < n do
87      if s <> 1 then
88        comp := InducedSubdigraph(gr_comp, [s .. n]);
89        comp := InducedSubdigraph(comp,
90                                  DigraphStronglyConnectedComponent(
     comp, 1));
91      fi;
92
93      if not IsEmptyDigraph(comp) then
94        for i in DigraphVertices(comp) do
95          blocked[i] := false;
96          B[i] := [];
97        od;
98        CIRCUIT(1, comp);
```

```
99        fi;
100       s := s + 1;
101     od;
102   od;
103   loops := List(DigraphLoops(digraph), x -> [x]);
104   return Concatenation(loops, out);
105 end);
```

Chapter 4

Spectral Theory for Tropical Matrices

The spectral radius of a matrix is equal to the supremum over the set of eigenvalues. Our first objective is to discuss the spectral radius for tropical, or max-plus, matrices and present a method for its efficient computation (see [4], [7]). Thematically, our approach draws on the max-plus precedence digraph (see chapter 2). Secondly, we consider a method for computing the eigenvector corresponding to the eigenvalue corresponding to spectral radius. Finally, we draw on the developed methodology to consider normalising max-plus matrix semigroups that satisfy a condition based on the spectral radius. This process is instrumental in chapter 5.

4.1 The spectral radius

First, we recall the formal definition of the spectral radius. Let $A \in \mathbb{R}_{\max}^{n\times n}$. Then, the spectral radius of A, denoted $\rho(A)$, is

$$\rho(A) = \sup\{r \in \mathbb{R}_{\max} | \, \exists u \in \mathbb{R}_{\max}^{n} \setminus \{\overline{\mathbb{0}}\} : Au = ru\}, \tag{4.1}$$

where $\overline{\mathbb{0}}$ denotes the zero vector with respect to the max-plus algebra. A straightforward approach to calculating this value is non-trivial, and computationally expensive in high dimen-

sions. Thus, we adopt an approach based on the precedence graph as described in [4] and [7]. Recall from chapter 3 that a circuit is a path of vertices $(v_1, v_2, v_3, ...v_k)$ with $v_1 = v_k$. For $A \in \mathbb{R}_{\max}^{n\times n}$, we denote the cycle mean by $m_A(c)$. For some circuit $c = (v_1, v_2, v_3, ...v_k = v_1)$ in $\mathcal{P}(A)$ with edges $c_e = \{(v_1, v_2), ..., (v_{k-1}, v_k), (v_k, v_1)\}$, this is defined to be

$$m_A(c) = \left(\bigotimes_{(i,j)\in c_e} A_{ij} \right)^{\frac{1}{k}}, \tag{4.2}$$

recalling that A_{ij} the weight of the edge (i, j). Furthermore, note that conventional division by k is equivalent to raising to the power of $(\frac{1}{k})$ with respect to $\otimes$. Hence, 4.2 is analogous to the usual concept of an arithmetic mean, since $\otimes$ translates as addition, and then we divide in the traditional sense. Thus, expressing the above with respect to the standard operations we get

$$\left(\bigotimes_{(i,j)\in c_e} A_{ij} \right)^{\frac{1}{k}} = \sum_{(i,j)\in c_e} \frac{A_{ij}}{k}. \tag{4.3}$$

Next, for the purposes of computing the spectral radius, we introduce the maximum cycle mean of $\mathcal{P}(A)$. Denoted $M(A)$, this is naturally defined as

$$M(A) = \max_{c\in\mathcal{C}(A)} m_A(c), \tag{4.4}$$

where $\mathcal{C}(A)$ denotes the set of all possible circuits in $\mathcal{P}(A)$. The following lemma cited in [4] establishes an equivalence between the maximum cycle mean and the spectral radius. For brevity we must omit the proof, which can be found in [5].

Lemma 4.1.1. *Let $A \in \mathbb{R}_{\max}^{n\times n}$ for $n \in \mathbb{N}$. Then*

$$\rho(A) = M(A).$$

It is immediate from lemma 4.1.1 that the problem of computing the spectral radius is reduced to computing the maximum cycle mean of $\mathcal{P}(A)$. To do this, we derive an alternative

form that is readily computable. We have the following proposition based on [7], noting that for a matrix $A \in \mathbb{R}_{\max}^{n\times n}$, the trace of A, denoted trace(A), is

$$\text{trace}(A) = \bigoplus_{i=1}^{n} A_{ii} = \max_{1\leq i\leq n\}} A_{ii}.$$

Proposition 4.1.2. *Let $A \in \mathbb{R}_{\max}^{n\times n}$ for $n \in \mathbb{N}$. Then,*

$$M(A) = \max_{1\leq k\leq n} \text{trace}(A^k)^{\frac{1}{k}} = \bigoplus_{1\leq k\leq n} \text{trace}(A^k)^{\frac{1}{k}}.$$

Proof. By definition $M(A) = \max_{c\in\mathcal{C}(A)} m_A(c)$. We can simplify this further using a standard result analogous to that used in chapter 1. In particular, for $k \in \mathbb{N}$, $(A^k)_{ij}$ is the maximum over the sum of the labels of the length k directed paths from vertex j to i in $\mathcal{P}(A)$. It follows that

$$\begin{aligned}
M(A) &= \max_{c\in\mathcal{C}(A)} m_A(c) \\
&= \max_{k\in\mathbb{N}} \left(\frac{1}{k} \max_{1\leq i\leq n} (A^k)_{ii} \right) \\
&= \max_{k\in\mathbb{N}} \bigoplus_{i=1}^{n} \frac{(A^k)_{ii}}{k} \\
&= \max_{k\in\mathbb{N}} \frac{\text{trace}(A^k)}{k} \\
&= \max_{k\in\mathbb{N}} \text{trace}(A^k)^{\frac{1}{k}} \\
&= \bigoplus_{1\leq k\leq n} \text{trace}(A^k)^{\frac{1}{k}},
\end{aligned}$$

recalling that division by k correspond to raising to the $\frac{1}{k}$ power with respect to $\otimes$. From [7], it suffices to consider simple circuits. Hence $M(A) = \max_{1\leq k\leq n} \text{trace}(A^k)^{\frac{1}{k}}$, as required. □

To conclude, we present the following GAP code based on proposition 4.1.2 and lemma 4.1.1, which computes the spectral radius of a max-plus matrix.

```
InstallMethod(SpectralRadius, "for a max–plus matrix",
[IsMaxPlusMatrix],
function(mat)
  local dim, cm, mk, k, max;

  # Check for –infinity case
  if DigraphAllSimpleCircuits(UnweightedPrecedenceDigraph(mat)) = []
   then
    return –infinity;
  fi;

  # Calculate the maximum cycle mean
  dim := Length(AsList(mat)[1]);
  cm := [];
  mk := mat;
  for k in [1 .. dim] do
    max := Maximum(List([1 .. dim], x –> mk[x][x]));
    if max <> –infinity then
      Add(cm, max/k);
    else
      Add(cm, max);
    fi;
    mk := mk * mat;
  od;

  return Maximum(cm);
end);
```

4.2 Radial eigenvectors

Of particular interest is an eigenvector corresponding to the spectral radius of a max-plus matrix. It is clear that there exist only finitely many simple circuits in a digraph with a finite vertex set. Hence, such an eigenvector exists, since the value of the supremum in the definition of the spectral radius is realised by some eigenvalue. Consequently, despite using the supremum operator for consistency with the literature, for our purposes it could be replaced with the maximum operator. Formally, we wish to find the eigenvector $u \in \mathbb{R}_{\max}^n \setminus \{\overline{\mathbb{0}}\}$ realising the supremum (or maximum) given by

$$\rho(A) = \sup\{r \in \mathbb{R}_{\max} | \, \exists u \in \mathbb{R}_{\max}^n \setminus \{\overline{\mathbb{0}}\} : Au = ru\}. \tag{4.5}$$

Naturally, we refer to this as a *radial* eigenvector. In regards to the order problem, we require a method for computing a radial eigenvector in the case that the spectral radius is zero. However, we first develop the theory in the slightly more general setting for non-positive spectral radii, mirroring [5]. Our case of interest then immediately follows.

Preceding the main result of this section, we require several definitions. Recall from the previous section that the spectral radius corresponds to the maximum cycle mean. A cycle which realises this maximum is known as a critical cycle, and is said to contain critical vertices. For a max-plus matrix A and a cycle c in $\mathcal{P}(A)$, these are denoted $V^c(A)$. Furthermore, for $A \in \mathbb{R}_{\max}^{n \times n}$ and $\lambda \in \mathbb{R}$, define the max-plus matrix A_λ by

$$(A_\lambda)_{ij} = A_{ij} - \lambda. \tag{4.6}$$

Next, we define A^+ as

$$\bigoplus_{k=1}^{\infty} A^k, \tag{4.7}$$

and the related matrix A^* as

$$A^* = I \oplus A^+ = \bigoplus_{k=0}^{\infty} A^k, \tag{4.8}$$

where I denotes the $n \times n$ max-plus identity matrix. Note that 4.8 holds since $A^0 = I$. Finally, for a vertex v of $\mathcal{P}(A)$, we denote the column of A corresponding to v as $[A]_v$.

To support the our main result we require the following lemma. This establishes a required conditions for the simple computation of A^+. In general, due to the infinite sum, the computation may not terminate and be intractable, or may require a more sophisticated analysis.

Lemma 4.2.1. *Let $A \in \mathbb{R}_{\max}^{n\times n}$. If $\rho(A) \leq 0$, then A^+ exists and is given by*

$$A^+ = \bigoplus_{k=1}^{2n} A^k. \tag{4.9}$$

Proof. It follows immediately from the definition of A^+ that

$$(A^+)_{ji} \geq \max\{(A^k)_{ji} : 1 \leq k \leq 2n\} \tag{4.10}$$

It remains to show that $(A^+)_{ji} \leq \max\{(A^k)_{ji} : 1 \leq k \leq 2n\}$. A path p of length $k > n$ in $\mathcal{P}(A)$ from i to j must visit at least one vertex multiple times. Hence, such a path contains at least one simple circuit and a path p' of length at most n from i to j. Since $\rho(A) = 0$, the maximum weight of a circuit from i is 0. If the maximum weight corresponding to $(A^+)_{ji}$ occurs on p' then we are done, since p is a path of length less than n. Assume it occurs on some simple circuit c. The path constructed by appending c to p' has length at most $2n$. This

yields $(A^+)_{ji} \leq \max\{(A^k)_{ji} : 1 \leq k \leq 2n\}$, as required. □

These definitions prepare us for a key result from [5], which forms the basis of an algorithm for the efficient computation of a radial eigenvector. The proof is technical and requires further auxiliary results, forcing its omission. See [5] for details.

Proposition 4.2.2. *Let $A \in \mathbb{R}_{\max}^{n\times n}$. Suppose $c \in \mathcal{C}$ is a circuit in $\mathcal{P}(A)$ of cycle mean $\rho(A)$. Then, for any $v \in V^c(A)$, the column $[A^*_{\rho(A)}]_v$ is an eigenvector corresponding to the eigenvalue $\rho(A)$.*

In the context of the order problem, we require a method to compute a radial eigenvector for a max-plus matrix A satisfying $\rho(A) = 0$ (see chapter 5). To conclude, we present the following GAP code that achieves this through the use of lemma 4.2.1 and proposition 4.2.2.

```
InstallMethod(RadialEigenvector,
"for a max–plus matrix",
[IsMaxPlusMatrix],

function(m)
  local dim, i, j, k, mplus, mstar, diag, crit;
  dim := Length(AsList(m)[1]);

  # Method valid for SpectralRadius(m) = 0
  if SpectralRadius(m) <> 0 then
    TryNextMethod();
  fi;

  mplus := List([1..dim], i -> List([1..dim], j ->
      Maximum(List([1 .. 2*dim], k -> AsList(m^k)[i][j]))));
```

```
16
17    mstar := mplus;
18    for i in [1..dim] do
19      mstar[i][i] := Maximum(mstar[i][i], 0);
20    od;
21
22    crit := false;
23    k := 1;
24    while crit = false do
25      diag := List([1..dim], i -> AsList(m^k)[i][i]);
26      if 0 in diag then
27        crit := Position(diag, 0);
28      fi;
29    k := k + 1;
30    od;
31    return List([1..dim], i -> AsList(mstar)[i][crit])
32  end);
```

4.3 Normalisation

In this section we consider a normalisation applicable for certain max-plus matrix semigroups that satisfy a property related to the spectral radius. To begin, we establish the norm of interest in the following lemma.

Lemma 4.3.1. *Let $A \in \mathbb{R}_{\max}^{n\times n}$. The function $\|\cdot\|$ defined by*

$$\|A\| = \bigoplus_{i,j} A_{ij} = \sup_{i,j} A_{ij} \tag{4.11}$$

is a norm on $\mathbb{R}_{\max}^{n\times n}$.

Proof. Suppose that $\|A\| = 0$ Then $\sup_{i,j} A_{ij} = 0$, from which it immediately follows that $A = Z$, the max-plus zero vector, as required. Secondly, observe that for $\lambda \in \mathbb{R}_{\max}$ we have

$$\begin{aligned} \|\lambda \otimes A\| &= \sup_{i,j} \lambda \otimes A_{ij} \\ &= \sup_{i,j} \lambda + A_{ij} \\ &= \lambda + \sup_{i,j} A_{ij} \\ &= \lambda \otimes \sup_{i,j} A_{ij} \\ &= \lambda \otimes \|A\| . \end{aligned} \tag{4.12}$$

Finally, consider

$$\begin{aligned} \|A \oplus B\| &= \sup_{i,j} A_{ij} \oplus \sup_{i,j} B_{ij} \\ &= \sup_{i,j} \max\{A_{ij}, B_{ij}\} \\ &\leq \max\{\sup_{i,j} A_{ij}, \sup_{i,j} B_{ij}\} \\ &= \sup_{i,j} A_{ij} \oplus \sup_{i,j} B_{ij} \\ &= \|A\| \oplus \|B\| , \end{aligned} \tag{4.13}$$

as required. □

Next, let $A_i \in \mathbb{R}_{\max}^{n\times n}$ for $i \in \{1, , ..., n\}$ and $S = \langle A_1, A_2, ..., A_k\rangle$. Furthermore, define $M = \bigoplus_{i=1}^{n} A_i$. In the case that $\rho(M) = 0$, we say that S is *normalised* if and only if

$$\sup\{\|A\| \,|A \in S\} = 0. \tag{4.14}$$

This naturally leads to the definition of $\mathbb{R}^-_{\max} = \{x \in \mathbb{R}_{\max} | x \leq 0\}$ and the corresponding set of matrices $(\mathbb{R}^-_{\max})^{n\times n}$. Clearly, if S is normalised then $S \subseteq (\mathbb{R}^-_{\max})^{n\times n}$. The following proposition proves such a normalisation can be achieved, and suggests an algorithm for the procedure by utilising the radial eigenvector introduced in section 4.2.

Proposition 4.3.2. *Let $S = S = \langle A_1, A_2, ..., A_k\rangle$ and $M = \bigoplus_{i=1}^{n} A_i$. If $\rho(M) = 0$, then there exists a diagonal matrix D such that DSD^{-1} is normalised.*

Proof. Observe that $\rho(M) = 0$ implies that 0 is an eigenvalue of M. Thus, the equation $Mu = 0 \otimes u = u$ has a solution $u \in \mathbb{R}^n_{\max}$, which is the radial eigenvector. Consider the diagonal matrix $D(u)$. By direct calculation we have

$$\begin{aligned}
\left\|D^{-1}MD\right\| &= \bigoplus_{i,j} (D^{-1}MD)_{ij} \\
&= \bigoplus_{i} \bigoplus_{j} (D^{-1}MD)_{ij} \\
&= \bigoplus_{i} \bigoplus_{j} u_i^{-1} M_{ij} u_j \\
&= \bigoplus_{i} u_i^{-1} u_i \\
&= \bigoplus_{i} -u_i \otimes u_i \\
&= \bigoplus_{i} -u_i + u_i \\
&= 0.
\end{aligned} \tag{4.15}$$

It follows that, for each generator A_i of S, the following inequality holds

$$\left\|D^{-1}A_iD\right\| \leq \left\|D^{-1}MD\right\| = 0, \tag{4.16}$$

since $\|A_i\| \leq \|M\|$. Hence

$$S' = D^{-1}SD = \langle D^{-1}A_1D, ..., D^{-1}A_kD\rangle \subseteq (\mathbb{R}^-_{\max})^{n\times n}. \tag{4.17}$$

It remains to show that there exists a matrix $A \in D^{-1}SD$ that satisfies $\|A\| = 0$. Since $\rho(M) = 0$, there exists a k, i such that $(M^k)_{ii} = 0$ by proposition 4.1.2. Hence $(\bigoplus_j A_j)^k_{ii} = 0$. In particular, there exists a j such that $(A_j^k)_{ii} = 0$, noting that $A_j^k \in S$. We show that $(D^{-1}A_j^kD)_{ii} = 0$. Observe that

$$(A_j^kD)_{ii} \geq 0 \otimes u_i = 0 + u_i = u_i, \tag{4.18}$$

and so

$$(D^{-1}A_j^kD)_{ii} \geq -u_i \otimes u_i = -u_i + u_i = 0. \tag{4.19}$$

Thus, since $S' \subseteq (\mathbb{R}^-_{\max})^{n\times n}$ by 4.17, we have that $(D^{-1}A_j^kD)_{ii} = 0$, as required. In particular, it follows that

$$0 \leq \left\|D^{-1}A_j^kD\right\| \leq 0. \tag{4.20}$$

Hence $\left\|D^{-1}A_j^kD\right\| = 0$ and thus S' is normalised, which completes the proof. □

To conclude, we present an implementation in GAP of the algorithm implied by the proof of proposition 4.3.2.

```
InstallMethod(NormalizeSemigroup,
"for a finitely generated semigroup of max–plus matrices",
[IsMaxPlusMatrixSemigroup],

function(S)
  local gens, dim, m, i, j, k, diag, critcol, d, ngens;
  gens := GeneratorsOfSemigroup(S);
  dim := Length(gens[1][1]);

  # Sum with respect to max–plus algebra of generators of S
```

```
  m := Matrix(IsMaxPlusMatrix, List([1..dim], i -> List([1..dim], j
   ->
      Maximum(List([1..Length(gens)], k -> gens[k][i][j])))));

  critcol := RadialEigenvector(m);
  d := List([1..dim], i -> List([1..dim], j -> -infinity));
  for i in [1..dim] do
    d[i][i] := critcol[i];
  od;
  d := Matrix(IsMaxPlusMatrix, d);

  ngens := List(gens, g -> InverseOp(d)*g*d);
  return Semigroup(ngens);
end);
```

Chapter 5

The Order Problem for Tropical Matrix Semigroups

A positive solution to the order problem for max-plus matrix semigroups is given in [4]. We review [4] and utilise methods from the preceding chapters to produce an effective implementation in GAP that decides the order problem.

To begin, let us recall the formal statement of the problem. Let $S = \langle A_1, A_2, ..., A_k \rangle$ for $A_i \in \mathbb{R}_{\max}^{n \times n}$ and $i = 1, ..., k$. The order problem asks if it can be determined in finite time whether S is finite. Observe that the order problem is clearly semi-decidable[1], since an enumeration of the elements terminates if S is finite. However, this method fails for the infinite case. Hence, the essence of the problem lies in the detection of infinite cardinality.

To begin, we state the main theorem established in [4], which builds on the work of Simon [11] and constitutes an affirmative answer to the Burnside problem[2] for max-plus matrix semigroups. The proof is involved and forms a substantial portion of [4], to which we direct the reader for brevity.

[1] Recall that a decision problem is semi-decidable if there exists an algorithm that answers positively in finite time.

[2] For the definition of the Burnside problem and torsion semigroups see chapter 0.

Theorem 5.0.3. *Let $S \subseteq \mathbb{R}_{\max}^{n\times n}$ be finitely generated. If S is torsion then S is finite.*

Theorem 5.0.3 immediately reduces the order problem to the torsion problem, which asks whether it is possible to determine in finite time whether a finitely generated semigroup of max-plus matrices is torsion.

5.1 The torsion problem

The reduction of the order problem to the torsion problem motivates this section. First, we present a positive solution from [4] to the torsion problem for max-plus matrix semigroups. In addition, we present an implementation in GAP that effectively decides the torsion problem for such semigroups.

Theorem 5.1.1. *Let S be a finitely generated semigroup of max-plus matrices, the torsion problem for S is decidable.*

In order to establish 5.1.1, we require a number of auxiliary results. First, consider the following theorem which translates a classical cyclicity result into the max-plus context. Numerous proofs exist, for example see [7] or [12].

Theorem 5.1.2. *Let $A \in \mathbb{R}_{\max}^{n\times n}$ be irreducible. Then there exists $N, c \geq 1$ such that for all $n \geq N$ we have*

$$A^{n+c} = \rho(A)^c A^n. \tag{5.1}$$

Theorem 5.1.2 has the following corollary [4], that provides an effective criterion for determining if a max-plus matrix is torsion.

Corollary 5.1.3. *Let $A \in \mathbb{R}_{\max}^{n\times n}$. Then the following are equivalent:*

1. *A is torsion,*
2. *for each irreducible block B of A, $\rho(B) \in \{-\infty, 0\}$.*

Proof. We prove the case where A consists of a single irreducible block, that is, A is irreducible. It is technical, but simple, to extend to the case where A is reducible, see [4]. Assume that A is torsion, then there exist p, q such that $A^p = A^q$ and

$$A^{r+k(q-p)} = A^r, \tag{5.2}$$

for all $r \geq p$ and $k \in \mathbb{N}$. In addition, by theorem 5.1.2 it follows that there exists $c, N \in \mathbb{N}$ such that

$$A^{n+c} = \rho(A)^c A^n, \tag{5.3}$$

for $n \geq N$. For all $t \in \mathbb{N}$, we prove that $A^{N+tc} = \rho(A)^{tc} A^N$ by induction on t. The base case is immediate. Assume the result holds for $t \leq k$. For $t+1$ we have

$$\begin{aligned} A^{N+(t+1)c} &= A^{(N+tc)+c} \\ &= \rho(A)^c A^{N+tc} \\ &= \rho(A)^c \rho(A)^{tc} A^N \\ &= \rho(A)^{(t+1)c} A^N. \end{aligned} \tag{5.4}$$

Thus, by induction, $A^{N+tc} = \rho(A)^{tc} A^N$ for all $t \in \mathbb{N}$. In particular, we have that $A^{N+(q-p)c} = \rho(A)^{(q-p)c} A^N$. By 5.2, we deduce that $A^{N+(q-p)c} = A^N$. Combining these results implies

$$A^N = \rho(A)^{(q-p)c} A^N \tag{5.5}$$

In order to reach a contradiction, assume $\rho(A) \notin \{0, -\infty\}$, then $\rho(A) = k \in \mathbb{R} \setminus \{0\}$. From 5.5 we deduce that $(A^N)_{11} = k + (A^N)_{11}$, a contradiction. Hence $\rho(A) \in \{0, -\infty\}$, since $\rho(A)$ exists and is an element of $\mathbb{R}_{\max}$.

Conversely, suppose that $\rho(A) \in \{0, -\infty\}$. We wish to show that A is torsion. First, consider the case where $\rho(A) = 0$. Observe that $\rho(A)^t = \rho(A)$ for all $t \in \mathbb{N}$. Hence,

theorem 5.1.2 implies there exists an $c, N \in \mathbb{N}$ such that

$$A^{n+c} = \rho(A)^c A^n = \rho(A) A^n = 0 \otimes A^n = A^n, \tag{5.6}$$

for $n \geq N$. Taking $n = N$, it immediately follows that A is torsion. Secondly, consider the case where $\rho(A) = -\infty$. Theorem 5.1.2 implies

$$(A^{n+c})_{ij} = (\rho(A)^c A^n)_{ij} = -\infty \otimes A^n_{ij} = -\infty, \tag{5.7}$$

for all i, j and $n \geq N$. In particular, $A^N = A^{N+k}$ for all $k \in \mathbb{N}$. Hence A is torsion, as required. □

Further theory is required to produce an analogous result for determining if an max-plus matrix semigroup is torsion, as opposed to a single matrix. Henceforth, let $S = \langle A_1, A_2, ..., A_k \rangle$ for $A_i \in \mathbb{R}^{n\times n}_{\max}$ and define $M = \bigoplus_{i=1}^{k} \mu(A_i)$. Our first proposition requires the following technical lemma.

Lemma 5.1.4. *Let $A \in \mathbb{R}^{n\times n}_{\max}$ and $k \geq 1$. Then*

$$\rho(A^k) = \rho(A)^k. \tag{5.8}$$

Proof. Observe that that by proposition 4.1.2 we have

$$\begin{aligned} \rho(A^k) &= \max_{1\leq j\leq n} \text{trace}((A^j)^k)^{\frac{1}{j}} \\ &= \max_{j\in\mathbb{N}} \text{trace}((A^j)^k)^{\frac{1}{j}} \\ &\leq \max_{j\in\mathbb{N}} \text{trace}(A^j)^{\frac{k}{j}} \\ &= \rho(A)^k. \end{aligned} \tag{5.9}$$

In addition, suppose that u is an eigenvalue of A, and so $Au = ru$ by definition. It follows that $A^k u = r^k$. Hence $\rho(A)^k$ is an eigenvalue of A^k. In particular, $\rho(A)^k \leq \rho(A^k)$, since

$\rho(A^k)$ is the supremum over all such eigenvalues. Combining these results, we deduce that $\rho(A^k) = \rho(A)^k$, as required. $\square$

Next, we prove a proposition that yields important insight on the constructed matrix $M = \bigoplus_{i=1}^{k} \mu(A_i)$. This relies on the introduction standard alternative notation for expressing the semigroup S. In particular, let $\Sigma = \{a_1, ..., a_k\}$ and $\mu : \Sigma^+ \to \mathbb{R}_{\max}^{n\times n}$ be a morphism such that $\mu(a_i) = A_i$ and $S = \mu(\Sigma^+)$. Note that we denote by Σ^+ the set of finite words over an arbitrary set Σ.

Proposition 5.1.5. *Observe $M = \bigoplus_i \mu(a_i)$. We have that*

$$\rho(M) = \bigoplus_{w\in\Sigma^+} \rho(\mu(w))^{\frac{1}{|w|}}, \tag{5.10}$$

where $|w|$ denotes the length of the word $w \in \Sigma^+$.

Proof. First, observe $\mu(a_i) \leq M$ for all i. Hence we have that

$$\bigoplus_{w\in\Sigma^+} \rho(\mu(w))^{\frac{1}{|w|}} \leq \bigoplus_{n\in\mathbb{N}} \rho(M^n)^{\frac{1}{n}} = \rho(M), \tag{5.11}$$

by an application of lemma 5.1.4. It remains to prove $\rho(M) \leq \bigoplus_{w\in\Sigma^+} \rho(\mu(w))^{\frac{1}{|w|}}$. By definition, there exists a sequence $i_1, i_2, ..., i_k \in \{1, ..., n\}$ corresponding to the maximum cycle mean equivalent to $\rho(M)$. Thus,

$$\rho(M)^k = M_{i_1 i_2} \otimes M_{i_2 i_3} \otimes ... \otimes M_{i_k i_1}. \tag{5.12}$$

By definition, observe that for all (i, j), we have that M_{ij} corresponds to $\mu(a_s)_{ij}$ for some s. Choose $s_l \in \{1, ..., n\}$ such that $M_{i_l i_l+1} = \mu(a_{s_l})_{i_l i_l+1}$. Then, for $w = a_{s_1} a_{s_2} ... a_{s_k}$, we

have

$$\begin{aligned}\rho(\mu(w)) &\geq \text{trace}(w) \\ &\geq \mu(w)_{i_1 i_1} \\ &\geq \mu(a_{s_1})_{i_1 i_2}\mu(a_{s_2})_{i_2 i_3}...\mu(a_{s_k})_{i_k i_1} \\ &= \rho(M)^k.\end{aligned} \tag{5.13}$$

It follows that

$$p(M) \leq \rho(\mu(w))^{\frac{1}{k}} = \rho(\mu(w))^{\frac{1}{|w|}} \leq \bigoplus_{w\in\Sigma^+} \rho(\mu(w))^{\frac{1}{|w|}}, \tag{5.14}$$

as required. □

Proposition 5.1.5 prepares us for the following theorem, which is instrumental. It provides an efficient method for detecting for torsion semigroups, and thus infinite order, in the vast majority of cases.

Theorem 5.1.6. *If S is torsion then $\rho(M) \in \{-\infty, 0\}$.*

Proof. By proposition 5.1.5, we have

$$\rho(M) = \rho(\mu(w))^{\frac{1}{|w|}} \tag{5.15}$$

for some $w \in \Sigma^+$. Since $\mu(w) \in S$, it is torsion. Consequently, since $\mu(w) \in \mathbb{R}_{\max}^{n\times n}$, then $\mu(w) \in \{-\infty, 0\}$ by corollary 5.1.3. Since $0^{\frac{1}{|w|}} = 0$ and $-\infty^{\frac{1}{|w|}} = -\infty$, we have $\rho(M) \in \{-\infty, 0\}$, as required. □

Notice that the utility of proposition 5.1.5 lied in showing $\rho(M)$ is witnessed by some $\mu(w) \in S$. On the basis of theorem 5.1.6, we divide our attention into the two cases where $\rho(M) = -\infty$ and $\rho(M) = 0$ respectively. The following proposition addresses the former case. Recall that $a \in S$ is nilpotent if $\exists\, n \in \mathbb{N}$ such that $a^n = Z$, where Z denotes the zero $n \times n$ max-plus matrix. Further, S is nilpotent if and only if there exists an $n \in \mathbb{N}$ such that $a^n = Z$ for all $a \in S$. Equivalently, we write $S^n = \{Z\}$.

Proposition 5.1.7. *$\rho(M) = -\infty$ if and only if S is nilpotent. In particular, if $\rho(M) = -\infty$ then S is torsion.*

Proof. First, observe that if S is nilpotent, then S is clearly torsion. Assume that $\rho(M) = -\infty$. By the equivalence of the spectral radius and the maximum cycle mean, M contains no circuits. In addition, $(M^n)_{ij} = -\infty$ for all (i, j), since a path of length n in M is either a complete circuit, or contains a circuit. Recall from the proof of 5.1.5 that $\mu(w) \leq M^n$ for all $w \in \Sigma^+$ of length n. Hence $(\mu(w))_{ij} = -\infty$ for all (i, j). Since w is an arbitrary word of length n, this implies $\mu(w) = Z$ for all w of length n (recall that Z denotes the matrix such that $Z_{ij} = -\infty$ for all (i, j)). Thus, for all $A \in S$, we have that $A^n = Z$, since a word w such that $\mu(w) = A$ satisfies $\mu(w^n) = A^n$ where $|w^n| \geq n$.

For the converse, suppose that S is nilpotent. In order to reach a contradiction, assume that M admits a circuit $c = (i_1 i_2 ... i_k i_1)$ of length k. Repeated traversal of c yields $(M^{mk})_{i_1 i_1} \neq -\infty$ for all $m \in \mathbb{N}$, since if $(M^{mk})_{i_1 i_1} = -\infty$ then no circuit of length mk exists from vertex i_1. This implies S contains non-zero products of arbitrary length. Hence S is not nilpotent, a contradiction. Thus M does not admit a circuit, which implies $\rho(M) = -\infty$, as required. □

For the remaining case, since $\rho(M) = 0$, the normalisation described in section 4.3 applies. Let the normalised subgroup $S' \subseteq (\mathbb{R}^-_{\max})^{n \times n}$ be constructed from S according to theorem 4.3.2. This is exploited in the following theorem.

Theorem 5.1.8. *Suppose $\rho(M) = 0$, then S is torsion if and only if S' is torsion.*

Proof. From theorem 4.3.2, recall that $S' = D^{-1}SD$ for a diagonal matrix D. First, assume S is torsion and let $A \in S'$ be arbitrary. By construction, $A = D^{-1}BD \in S'$ for some $B \in S$. By assumption B is torsion, there exists $p, q \in \mathbb{N}$ such that $B^p = B^q$. It follows that

$$A^p = D^{-1}B^pD = D^{-1}B^qD = A^q, \tag{5.16}$$

as required. For the converse, assume that S' is torsion and let $A \in S$ be arbitrary. By assumption, $B = D^{-1}AD \in S'$ is torsion. Thus, there exists $p, q \in \mathbb{N}$ such that $B^p = B^q$. Hence $D^{-1}A^pD = D^{-1}A^qD$. By cancellation, this implies $A^p = A^q$ and hence A is torsion, as required. □

Hence, we have reduced the torsion problem in the case that $\rho(M) = 1$ to subsemigroups of $(\mathbb{R}_{\max}^-)^{n\times n}$. To proceed, we must define a map given in cited in [4] from [11]. Define the function $\kappa : \mathbb{R}_{\max}^- \to \mathbb{R}_{\max}^-$ by

$$\kappa(x) = \begin{cases} 0 & \textit{if } x = 0 \\ -1 & \textit{if } -\infty \leq x < 0\,, \\ -\infty & \textit{if } x = -\infty \end{cases} \tag{5.17}$$

which we extend component-wise to $(\mathbb{R}_{\max}^-)^{n\times n}$. The following lemma highlights two key properties of this function.

Lemma 5.1.9. *The function $\kappa : \mathbb{R}_{\max}^- \to \mathbb{R}_{\max}^-$ satisfies*

$$\begin{aligned} \kappa(a \oplus a) &= \kappa(a) \oplus \kappa(a), \\ \kappa(a \otimes b) &= \kappa(\kappa(a) \otimes \kappa(b)). \end{aligned} \tag{5.18}$$

for all $a, b \in \mathbb{R}_{\max}^-$.

Proof. By direct computation, observe

$$\kappa(a \oplus b) = \kappa(\max\{a, b\}) = \begin{cases} \kappa(a) & \textit{if } a \geq b \\ \kappa(b) & \textit{if } a < b \end{cases} = \max\{\kappa(a), \kappa(b)\} \tag{5.19}$$

since $a \geq b$ implies $\kappa(a) \geq \kappa(b)$ and $a < b$ implies $\kappa(a) \leq \kappa(b)$. It is then immediate from

the definition that $\kappa(a \oplus b) = \kappa(a) \oplus \kappa(a)$, as required. Secondly, observe that

$$\kappa(a \otimes b) = \kappa(a + b) = \begin{cases} 0 & \textit{if } a = b = 0 \\ -1 & \textit{if } a = 0 \textit{ and } -\infty \leq b < 0 \\ -1 & \textit{if } b = 0 \textit{ and } -\infty \leq a < 0 \\ -1 & \textit{if } -\infty \leq a, b < 0 \\ -\infty & \textit{if } a = -\infty \textit{ or } b = -\infty \end{cases} \tag{5.20}$$

From this, direct computation in each case verifies $\kappa(a \otimes b) = \kappa(\kappa(a) \otimes \kappa(b))$, as required. □

It follows from lemma 5.1.9 that κ is a morphism into the three element semiring $\mathbb{R}_\kappa = \{0, -1, -\infty\}$ that is equipped with the operations $a \oplus_\kappa b = a \oplus b$ and $a \otimes_\kappa b = \kappa(a \otimes b)$. The next lemma clarifies multiplication for matrices over the semiring $\mathbb{R}_\kappa$.

Lemma 5.1.10. *Let $A, B \in \mathbb{R}_3$. Then*

$$A \otimes_\kappa B = \kappa(A \otimes B) \tag{5.21}$$

Proof. For a valid pair of indices (i, j), we have

$$(A \otimes_\kappa B)_{ij} = \bigoplus_{k \in \{1,\ldots,n\}} A_{ik} \otimes_\kappa B_{kj}, \tag{5.22}$$

since $a \oplus_\kappa b = a \oplus b$. We then have

$$\begin{aligned} \bigoplus_{k \in \{1,\ldots,n\}} A_{ik} \otimes_\kappa B_{kj} &= \bigoplus_{k \in \{1,\ldots,n\}} \kappa(A_{ik} \otimes B_{kj}) \\ &= \kappa\left(\bigoplus_{k \in \{1,\ldots,n\}} A_{ik} \otimes B_{kj} \right) \\ &= (\kappa(A \otimes B))_{ij}, \end{aligned} \tag{5.23}$$

by noting that $a \geq b$ implies $\kappa(a) \geq \kappa(b)$ and $a < b$ implies $\kappa(a) \leq \kappa(b)$. Hence

$$A \otimes_\kappa B = \kappa(A \otimes B), \tag{5.24}$$

as required. □

We are now prepared to state and prove the main result of this section. In the following, let $\iota : \kappa(\mathbb{R}_{\max}^-)^{n\times n} \to (\mathbb{R}_{\max}^-)^{n\times n}$ denote the set inclusion map.

Proposition 5.1.11. *Let $A \in (R_{\max}^-)^{n\times n}$. The following are equivalent*

1. *A is torsion,*
2. *$\iota \circ \kappa(A)$ is torsion,*
3. *for each non-zero irreducible block B of A, there exists a circuit of B composed only of arcs of weight $\mathbb{1}$.*

Proof.

$(1 \iff 3)$ First, assume A is torsion and let B be a non-zero irreducible block of A. By corollary 5.1.1, we have that $\rho(B) = \{0, -\infty\}$. However, if $\rho(B) = -\infty$ then $B_{ij} = -\infty$ for all (i, j) since B irreducible implies $\mathcal{P}(B)$ is strongly connected. Hence $\rho(B) = 0$. Observe a circuit in B has weight 0 then the weight of each edge in that circuit must also have weight 0, since $B_{ij} \leq 0$ for all (i, j) by hypothesis. Hence the result follows, since $\rho(B) = 0$ corresponds to a particular cycle mean in B.

Secondly, assume that for each irreducible block B of A, there exists a circuit of B composed only of arcs of weight 0. Then $\rho(B) \geq 0$. In addition, since $B_{ij} \leq 0$ for all (i, j), it follows $\rho(B) \leq 0$. Hence $\rho(B) = 0$ and A is torsion by corollary 5.1.1, since B was an arbitary irreducible block of A.

$(2 \iff 3)$ This is immediate, since A can be replaced by $\iota \circ \kappa(A)$ above. □

Proposition 5.1.11 enables us to decide the torsion problem in this case, since the semigroup $\kappa(S)$ is finite. Thus, we can use proposition 5.1.11 to deduce S is torsion if and only if $\iota(A)$ is torsion for all $A \in \kappa(S)$. This is easily verified in finite time by the third condition above. Hence, by means of proposition 5.1.11 and results throughout, an algorithm for deciding the torsion problem in all cases is immediate. Moreover, the existence of such an algorithm constitutes a proof of theorem 5.1.1.

Input $S = \langle A_1, A_2, ..., A_k \rangle$ for $A_i \in \mathbb{R}_{\max}^{n\times n}$.

Output $B \in \{\texttt{TRUE}, \texttt{FALSE}\}$.

Step 1 Compute $M = \bigoplus_{i=1}^{k} \mu(A_i)$.

Step 2 Compute $\rho(M)$:

- If $\rho(M) \notin \{-\infty, 0\}$ return `FALSE`,
- If $\rho(M) = -\infty$ return `TRUE`.

Step 3 Construct the normalised semigroup S'.

Step 4 Compute the finite semigroup $\kappa(S)$.

Step 5 For each $A \in \circ\kappa(S)$, if there exists a non zero irreducible block B of $\iota(A)$, such that there does not exist a circuit of B composed only of arcs of weight 0, return `FALSE`.

Step 6 Return `TRUE`.

We conclude by presenting an implementation of this algorithm in GAP for inclusion in the Semigroups package.

```
InstallMethod(IsTorsion,
"for a finitely generated max-plus matrix semigroup",
[IsMaxPlusMatrixSemigroup],

function(S)
  local gens, dim, m, rad, s, kappa, g, sim, growth, e, f, x,
   cycletest, zeroblock, t,
        irrblocks, ib, circ;

  gens := GeneratorsOfSemigroup(S);
  dim := Length(gens[1][1]);
  m := Matrix(IsMaxPlusMatrix, List([1..dim], i -> List([1..dim], j
   ->
      Maximum(List([1..Length(gens)], k -> gens[k][i][j])))));

  # Case: SpectralRadius = -infinity
  rad := SpectralRadius(m);
  if rad = -infinity then
    return true;
  else if rad <> 0 then
    return false;
  fi;
  fi;

  # Case: SpectralRadius = 0
  s := NormalizeSemigroup(S);
  gens := GeneratorsOfSemigroup(s);
```

```
26
27  kappa := function(A)
28    local B, dim, i ,j;
29    dim := Length(A[1]);
30    B := List([1..dim], i -> List([1..dim], j -> 0));
31    for i in [1..dim] do
32      for j in [1..dim] do
33        if A[i][j] = 0   then
34          B[i][j] := 0;
35        else if -infinity < A[i][j] and A[i][j] < 0  then
36          B[i][j] := -1;
37        else
38          B[i][j] := -infinity;
39        fi;
40        fi;
41      od;
42    od;
43    return Matrix(IsMaxPlusMatrix, B);
44  end;
45
46  # Compute elements kappa(S) w.r.t multiplication kappa(a \otimes b)
47  g := List(gens, x -> kappa(x));
48  # Optimised for dim <= 3
49  sim := EmptyPlist(3^9);
50  sim{[1..Length(g)]} := g;
51  growth := true;
52  while growth do;
```

```
    growth := false;
    for e in sim do
      for f in g do
        x := kappa(e*f);
         if not x in sim then
          Add(sim, x);
          growth := true;
        fi;
      od;
    od;
  od;

  # ib is global variable
  cycletest := function(c)
    local i;
    for i in [1..Length(c)] do
      if i < Length(c) then
        if ib[c[i]][c[i+1]] <> 0 then
          return true;
        fi;
      else
        if ib[c[i]][c[1]] <> 0 then
          return true;
        fi;
      fi;
    od;
    return false;
```

```
80  end;
81
82  zeroblock := function(blk)
83    local i, j;
84    for i in [1..Length(blk[1])] do
85      for j in [1..Length(blk[1])] do
86        if blk[i][j] <> -infinity then
87          return false;
88        fi;
89      od;
90    od;
91    return true;
92  end;
93
94  for t in sim do
95    irrblocks := IrreducibleBlocks(t);
96    for ib in irrblocks do
97      if not zeroblock(ib) then
98        circ := DigraphAllSimpleCircuits(UnweightedPrecedenceDigraph(
     ib));
99        if ForAll(circ, cycletest) then
100         return false;
101       fi;
102     fi;
103   od;
104 od;
105 return true;
```

```
106 end);
```

5.2 The order problem

Our overarching objective has been achieved. For completeness, we include the now trivial method in GAP that decides the order problem for max-plus matrix semigroups. The following is justified by the reduction given in theorem 5.0.3. In the next chapter, we'll see the effectiveness of this method with a number of tests.

```
1 InstallMethod(IsFinite,
2 "for a finitely generated max-plus matrix semigroup",
3 [IsMaxPlusMatrixSemigroup],
4
5 function(S)
6   return IsTorsion(S);
7 end);
```

Chapter 6

Summary and Testing

This project has culminated in an effective implementation in GAP that determines if a finitely generated natural or tropical matrix semigroup is finite. For the former case, an algorithm from [3] was utilised, which exploited a morphism to a finite semigroup of matrices over a three element semiring. In chapter 5, we saw a parallel approach for the tropical case. Moreover, this case centred on a reduction to the torsion problem given in [4]. We relied heavily on a method to compute the spectral radius of a max-plus matrix, by utilising the precedence digraph. Importantly, the spectral radius highlighted a particular case where a normalisation could be achieved via conjugation with a diagonal matrix. This, in turn, relied on theory enabling the computation of the radial eigenvector from [5].

To conclude, we verify the effectiveness of our implementation of an `IsFinite` function for max-plus matrices for a number of tests. These cover the four key cases, which arise depending on the value of the spectral radius for the matrix $M = \bigoplus_{i=1}^{k} \mu(A_i)$. The code used to run these tests can be found in the code archive that accompanies this thesis.

Example 6.0.1.

First, let

$$I = \begin{pmatrix} 0 & -3 \\ -2 & -10 \end{pmatrix}. \tag{6.1}$$

For $S = \langle I \rangle$, we ha $M = I$. Our test yielded:

```
### new method
gap> I := Matrix(IsMaxPlusMatrix,[[0,-3],[-2,-10]]);
gap> SpectralRadius(I);
0
gap> isfinite(Semigroup(I));
true


### enumeration via existing method.
gap> IsFinite(Semigroup(I));
semigroups++: enumerate
limit = 18446744073709551615
found 2 elements, 1 rules, max word length 2, finished!
true
```

Note that this covers the case where S is finite and $\rho(M) = 0$. Next, we consider

$$I = \begin{pmatrix} -\infty & 1 & -\infty \\ -\infty & -\infty & -\infty \\ -\infty & 1 & -\infty \end{pmatrix}, \tag{6.2}$$

and the new semigroup $S = \langle I \rangle$. Similarly, $M = I$, and we have:

```
### new method
gap> I := Matrix(IsMaxPlusMatrix, [[-infinity, 1, -infinity],
> [-infinity, -infinity, -infinity], [-infinity, 1, -infinity]]);
gap> SpectralRadius(I);
-infinity
gap> isfinite(Semigroup(I));
true


### enumeration via existing method
gap> IsFinite(Semigroup(I));
semigroups++: enumerate
limit = 18446744073709551615
found 2 elements, 1 rules, max word length 2, finished!
true
```

Note that this covers the case where S is finite and $\rho(M) = -\infty$. We now move on to the infinite case. First, we consider

$$I = \begin{pmatrix} 1 & -\infty & 2 \\ -2 & 4 & -\infty \\ 1 & 0 & 3 \end{pmatrix}, \tag{6.3}$$

and the new semigroup $S = \langle I \rangle$. Again, $M = I$, and we have:

```
### new method
gap> I := Matrix(IsMaxPlusMatrix, [[1, -infinity, 2],
> [-2, 4, -infinity], [1, 0, 3]]);
gap> SpectralRadius(I);
4
gap> isfinite(Semigroup(I));
```

```
false

### enumeration via existing method: appears not to terminate.
### > 150000 elements found.
```

In this case, our algorithm claims S is infinite and $\rho(M) = 4$. This is supported by theory and supported by our test, since an enumeration of S using existing methods yielded over 10^5 elements. Finally, we consider

$$I = \begin{pmatrix} 0 & -\infty & -1 \\ 0 & -\infty & -4 \\ -\infty & 0 & -\infty \end{pmatrix} \tag{6.4}$$

$$J = \begin{pmatrix} -2 & -2 & 0 \\ 0 & -\infty & -3 \\ -5 & 0 & -\infty \end{pmatrix} \tag{6.5}$$

$$K = \begin{pmatrix} -\infty & -\infty & -\infty \\ 0 & 0 & -\infty \\ -5 & 0 & -\infty \end{pmatrix} \tag{6.6}$$

and the semigroup $S = \langle I, J, K \rangle$. By direct computation, we have

$$M = \begin{pmatrix} 0 & -2 & 0 \\ 0 & 0 & -3 \\ -5 & 0 & -\infty \end{pmatrix}. \tag{6.7}$$

Our test yielded the following:

```
### new method
gap> I := Matrix(IsMaxPlusMatrix, [[0, -infinity, -1],[0, -infinity, -4],
>  [-infinity, 0, -infinity]]);
```

```
gap> J := Matrix(IsMaxPlusMatrix, [[-2, -2, 0],[0, -infinity, -3],
>  [-5, 0, -infinity]]);
gap> K := Matrix(IsMaxPlusMatrix, [[-infinity, -infinity,
> -infinity],[0, 0, -infinity],[-5, 0, -infinity]]);
gap> M := Matrix(IsMaxPlusMatrix, [[0, -2, 0],[0, 0, -3],
>  [-5, 0, -infinity]]);
gap> SpectralRadius(M);
0
gap> isfinite(Semigroup(I,J,K));
false

### enumeration via existing method: appears not to terminate.
### > 150000 elements found.
```

This covers our final key case, where S is infinite and $\rho(M) = 0$. As before, evidence provided by a simple enumeration supports the algorithms output.

The above constitutes only preliminary testing. Moving on, the author recommends large scale testing be done that utilises an automated procedure for max-plus matrix semigroup generation. This will provide further evidence for the validity of this implementation. In addition, further work should be done to optimise these implementations and improve runtime. That said, runtime was almost instant for the above tests. Finally, it would be valuable to analyse the complexity of this algorithm. This would provide insight on its utility as instance size increases.

Bibliography

[1] The GAP Group. *GAP -- Groups, Algorithms, and Programming, Version 4.8.3*, 2016.

[2] J.D. Mitchell et al. Semigroups - GAP package, version 2.7.4, 2016. `http://www.gap-system.org/Packages/semigroups.html`.

[3] I. Simon A. Mandel. On Finite Semigroups of Matrices. *Theoretical Computer Science*, 5:101--111, 1977.

[4] S. Gaubert. On the Burnside problem for Semigroups of Matrices over the (max, +) Algebra. *Semigroup Forum*, 52:271--292, 1996.

[5] K.G. Farlow. Max-Plus Algebra. Master's thesis, Virginia Polytechnic Institute and State University, United States, 2009.

[6] D.B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77--84, 1975.

[7] G. J. Olsder J.P. Quadrat F. Baccelli, G. Cohen. *Synchronisation and Linearity: An Algebra for Discrete Event Systems*. Wiley, 1992.

[8] J. De Beule, J. Jonušas, J. D. Mitchell, M. Torpey, and W. Wilson. *Digraphs - GAP package, Version 0.5*, March 2016.

[9] J.C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Comm. ACM*, 13:722--726, 1970.

[10] R. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM J. Comput.*, 2:211--216, 1973.

[11] I. Simon. Limited subsets of the free monoid. *Proc. of the 19th Annual Symposium on Foundations of Computer Science*, pages 143--150, 1978. IEEE.

[12] P. I. Dudnikov and S. N. Samborski. Endomorphisms of semimodules over semirings with an idempotent operation. *Mathematics of the USSR-Izvestiya*, 38(1):91, 1992.