SOME ALGORITHMIC PROBLEMS IN MONOIDS OF BOOLEAN MATRICES

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2018

Peter J Fenner
School of Mathematics
# Contents

Abstract 4

Declaration 5

Copyright Statement 6

Acknowledgements 8

1 Introduction 9
   1.1 Semigroups .................................................. 9
   1.2 Computational Complexity .................................. 15
   1.3 Structure of the Thesis .................................... 19

2 Boolean Monoids 20

3 Reflexive Monoids 38

4 Upper Triangular Monoids 52

5 Unitriangular Monoids 57

6 Gossip Monoids 62

7 One-Directional Gossip Monoids 98
A Boolean matrix is a matrix with elements from the Boolean semiring $(\{0, 1\}, +, \cdot)$, where the addition and multiplication are as usual with the exception that $1 + 1 = 1$. In this thesis we study eight classes of monoids whose elements are Boolean matrices.

Green’s relations are five equivalence relations and three pre-orders which are defined on an arbitrary monoid $M$ and describe much of its structure. In the monoids we consider the equivalence relations are uninteresting - and in most cases completely trivial - but the pre-orders are not and play a vital part in understanding the structure of the monoids. Each of the three pre-orders in each of the eight classes of monoids can be viewed as a computational decision problem: given two elements of the monoid, are they related by the pre-order? The main focus of this thesis is determining the computational complexity of each of these twenty-four decision problems, which we successfully do for all but one.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University
IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s Policy on Presentation of Theses.
Acknowledgements

I would like to thank my supervisors, Prof. Mark Kambites and Dr Marianne Johnson for their guidance and support throughout the course of my PhD.

I thank EPSRC for financially supporting my research through a Doctoral Training Award.

I also thank the non-academic staff at the University of Manchester, especially Mrs Anna Bigland, for the incredibly useful assistance they have provided on several occasions.

I would like to thank the people with whom I have shared an office at university, in particular Dominic Hipwood, Matthew Taylor, and Tom Withers, for their friendship and for giving advice when it was needed. I would also like to thank friends from outside of the university, Phil Carty, Andrew Quigley, and especially Curt Anderson, who were there to listen to me rant whenever academic life wasn’t going so well.

Most importantly, I wish to thank my wife Katy and my daughters Poppy and Daisy for their emotional support, and for being by my side through all the difficulties of the last few years. We’ve been through a lot since I started my PhD, but we’ve been through it all together.
Chapter 1

Introduction

This thesis looks at several monoids whose elements are Boolean matrices. Boolean matrices can be used to represent directed graphs or binary relations on a finite set, and have been studied extensively in these various guises. The monoids we consider have each been studied elsewhere, some more than others, for various reasons. Some represent particular types of binary relations, some relate to the process of sharing information over a network, and others arise naturally from consideration of other algebraic structures. We will begin by briefly covering the aspects of semigroup theory used in this thesis. For a more detailed exposition with proofs of claimed elementary facts we direct the reader to [6].

1.1 Semigroups

Semigroups are a generalisation of groups which were first considered at the start of the 20th century. The first definition of a semigroup was given in 1904 by de Séguier [9] although his definition was different to the modern definition. Dickson [10] showed that de Séguier’s definition was equivalent to: a semigroup is a set $S$ with an associative binary operation such that for any $a, x, x' \in S$, if $ax = ax'$ or $xa = x'a$ then $x = x'$. In modern terminology this is known as a cancellative semigroup, and
the modern definition of a semigroup omits the cancellative property:

**Definition 1.1: Semigroup**

A *semigroup* is a set $S$ together with an associative operation.

This generalises the definition of a group in two ways:

- a semigroup need not have an identity element,
- elements of a semigroup need not have inverses.

The second generalisation obviously follows from the first, since the definition of an inverse relies on there being an identity element. We can, however, consider structures which generalise in the second way but not the first. These structures are known as *monoids*, a term which seems to have originated with Bourbaki [3]:

**Definition 1.2: Monoid**

A *monoid* is a set $M$ together with an associative operation and a distinguished element $1$, known as the identity, such that for all $m \in M$, $1m = m1 = m$.

In a similar way, we can consider a structure which is like a ring (not necessarily commutative) except elements need not have additive inverses:

**Definition 1.3: Semiring**

A *semiring* is a set $S$ with two associative operations $+$ and $\cdot$ such that:

- $S$ forms a commutative monoid under $+$, with identity element $0$,
- $S$ forms a monoid under $\cdot$, with identity element $1$,
- $\cdot$ distributes over $+$,
- $0 \cdot s = s \cdot 0 = 0$ for all $s \in S$.

An early consideration of semirings, possibly the first, can be found in [33].

Much as a field is a ring in which every element has a multiplicative inverse, we can define a semifield as a semiring in which every element has a multiplicative inverse:
Definition 1.4: Semifield
A semifield is a semiring in which each element has a multiplicative inverse, that is for each $s \in S$ there exists an element $s^{-1} \in S$ such that $ss^{-1} = s^{-1}s = 1$.

The semiring which we are primarily concerned with in this thesis is an example of a semifield. It is known as the Boolean semiring:

Definition 1.5: Boolean Semiring
Let $\mathbb{B} = (\{0, 1\}, +, \cdot)$ be the semiring with operations $+$ and $\cdot$ defined as usual except that $1 + 1 = 1$. We call this the Boolean semiring.

Much of the information about the structure of a semigroup can be described by Green’s relations. These are five equivalence relations and three pre-orders which were defined by Green in [14]. They apply to any monoid, and can more generally be applied to other semigroups by first adding a new element to act as an identity, although we will only be applying them to monoids in this thesis.

Green’s equivalence relations on a monoid $M$ are:

- $x \mathcal{L} y \iff Mx = My$,
- $x \mathcal{R} y \iff xM = yM$,
- $x \mathcal{J} y \iff MxM = MyM$,
- $x \mathcal{H} y \iff x \mathcal{L} y$ and $x \mathcal{R} y$,
- $x \mathcal{D} y \iff \exists z \in M, x \mathcal{L} z$ and $z \mathcal{R} y$.

It is clear that the first four of these are equivalence relations. To see that the last is an equivalence relation we refer the reader to [14].

The pre-orders are:

- $x \leq_{\mathcal{L}} y \iff Mx \subseteq My$, or equivalently, $\exists u \in M, x = uy$,
In this thesis we will only be looking at finite monoids. In a finite monoid
\(\mathcal{J}\)-equivalence and \(\mathcal{D}\)-equivalence coincide, as shown in [14].

If in the monoid \(M\) the equivalence relation \(\mathcal{L}\) is equality, then \(M\) is said to
be \(\mathcal{L}\)-trivial. \(\mathcal{R}\)-triviality, \(\mathcal{J}\)-triviality, \(\mathcal{H}\)-triviality, and \(\mathcal{D}\)-triviality are defined
analogously. In most of the monoids considered in this thesis all five equivalence
relations turn out to be trivial. In this case all of the structural information
described by Green’s relations is done so by the three pre-orders.

We will be looking at eight classes of monoids, each of which contains Boolean
matrices as its elements. We will briefly define them here, but we will give
background for each class of monoids in its own section.

Definition 1.6: Boolean Monoid of rank \(n\)
We write \(B_n\) for the monoid of \(n \times n\) matrices over \(B\). This is called the Boolean
monoid of rank \(n\).

Definition 1.7: Reflexive Monoid of rank \(n\)
The reflexive monoid of rank \(n\), denoted \(R_n\), is the submonoid of Boolean matrices
with 1s on the diagonal:

\[
R_n = \{ A \in B_n : \forall i \leq n, a_{i,i} = 1 \}.
\]

Definition 1.8: Upper Triangular Monoid of rank \(n\)
The upper triangular monoid of rank \(n\), denoted \(UT_n\), is the submonoid of Boolean
matrices with 0s below the diagonal:

\[
UT_n = \{ A \in B_n : i > j \implies a_{i,j} = 0 \}.
\]
**Definition 1.9: Unitriangular Monoid of rank $n$**

The unitriangular monoid of rank $n$, denoted $U_n$, is the submonoid of Boolean matrices with 0s below the diagonal and 1s on the diagonal:

$$U_n = R_n \cap UT_n.$$ 

**Definition 1.10: Gossip Monoid of rank $n$**

We define an type of element of $B_n$ called a phone call matrix as follows: Given $a, b \leq n$, let $C[a, b]$ be the $n \times n$ Boolean matrix such that

$$c[a, b]_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } \{i, j\} = \{a, b\}, \\
0 & \text{otherwise}.
\end{cases}$$

The gossip monoid of rank $n$, denoted $G_n$, is the monoid generated by all $n \times n$ phone call matrices:

$$G_n = \langle \{C[a, b] : a, b \leq n\} \rangle.$$ 

**Definition 1.11: One-Directional Gossip Monoid of rank $n$**

We define an type of element of $B_n$ called a message matrix as follows: Given $a, b \leq n$, let $M[a, b]$ be the $n \times n$ Boolean matrix such that

$$m[a, b]_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } (i, j) = (a, b), \\
0 & \text{otherwise}.
\end{cases}$$

The one-directional gossip monoid of rank $n$, denoted $\bar{G}_n$, is the monoid generated by all $n \times n$ message matrices:

$$\bar{G}_n = \langle \{M[a, b] : a, b \leq n\} \rangle.$$ 

**Definition 1.12: Catalan Monoid of rank $n$**

For each $1 \leq k \leq n - 1$, we define the $n \times n$ matrix $\delta_k$ by

$$(\delta_k)_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } (i, j) = (k, k + 1), \\
0 & \text{otherwise}.
\end{cases}$$
In other words, \( \delta_k = M[k, k + 1] \).

The Catalan monoid of rank \( n \), denoted \( C_n \), is the monoid generated by the matrices \( \delta_k \) for \( 1 \leq k \leq n - 1 \):

\[
C_n = \langle \{\delta_k : 1 \leq k \leq n - 1\} \rangle.
\]

**Definition 1.13: Double Catalan Monoid of rank \( n \)**

For each \( 1 \leq k \leq n - 1 \), we define the \( n \times n \) matrix \( \varepsilon_k \) by

\[
(\varepsilon_k)_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } \{i, j\} = \{k, k + 1\}, \\
0 & \text{otherwise.} 
\end{cases}
\]

In other words, \( \varepsilon_k = C[k, k + 1] \).

The double Catalan monoid of rank \( n \), denoted \( DC_n \), is the monoid generated by the matrices \( \varepsilon_k \) for \( 1 \leq k \leq n - 1 \):

\[
DC_n = \langle \{\varepsilon_k : 1 \leq k \leq n - 1\} \rangle.
\]

Many of these monoids are submonoids of each other. The following diagram shows the relationships between them:

\[
C_n \subseteq U_n \subseteq UT_n \subseteq B_n \\
\bigcap \quad \bigcup
\]

\[
DC_n \subseteq G_n \subseteq \bar{G}_n \subseteq R_n
\]

The only inclusion which is perhaps not obvious is \( U_n \subseteq \bar{G}_n \). This will be proved in the section on one-directional gossip monoids in Theorem 7.5.

As previously mentioned, in most of these monoids (all of them except the Boolean monoids and upper triangular monoids) all of Green’s equivalence relations are trivial, so any structure described through Green’s relations is done entirely through the pre-orders. In this thesis we look at the computational problem of determining, for a pair of monoid elements \( a, b \in M \), whether or not \( a \leq_L b \), \( a \leq_R b \),
or $a \leq_J b$. Specifically we look at the computational complexity of these problems in each of the above monoids. In the following paragraphs we provide a very brief (and non-rigorous) introduction to computational complexity.

### 1.2 Computational Complexity

Computational complexity theory looks at decision problems (problems which have a yes/no answer depending on an input) and attempts to determine how difficult they are to solve on a computational device such as a Turing machine. There are two types of Turing machine, deterministic and non-deterministic, and we categorise problems based on how long it takes to solve them on these two types of machine in terms of the size of the input.

Turing machines are theoretical tape-based computational devices, but for the purposes of this thesis we can think of a deterministic Turing machine as a computer with a single processor and a non-deterministic Turing machine as a computer with an infinite number of processors. The amount of time taken for a Turing machine to solve a problem is the number of successive operations the Turing machine must perform to arrive at the solution. For a non-deterministic Turing machine, any number of operations can be performed simultaneously in a single unit of time, but if one operation relies on the result of another operation then they must still be performed one after the other.

Strictly speaking, the size of the input is the amount of space required to store the input data on tape, but in practice we can use any definition of size for which there exists a polynomial $p$ such that the space needed to store any input of size $n$ is bounded above by $p(n)$, and generally any intuitively sensible notion of size will suffice. Since we are thinking of Turing machines as computers we can think in terms of bytes rather than space on tape. In the case of square matrices we take
the size to be the order of the matrix.

In this thesis we will be categorising decision problems as either P or NP-complete. A problem is P if there is a polynomial \( p \) such that, given any input of size \( n \), a deterministic Turing machine can solve the problem for that input in time no greater than \( p(n) \). A problem is NP-complete if it is both NP and NP-hard: it is NP if there is a polynomial \( p \) such that, given any input of size \( n \), a non-deterministic Turing machine can solve the problem for that input in time no greater than \( p(n) \). To describe NP-hardness we need the concept of a polynomial time reduction.

A polynomial time many-one reduction (often simply known as a polynomial time reduction) from problem \( A \) to problem \( B \) is a mapping \( \phi \) from the set of valid inputs to problem \( A \) to the set of valid inputs to problem \( B \) such that:

a) problem \( A \) is true for input \( x \) if and only if problem \( B \) is true for input \( \phi(x) \),

b) a deterministic Turing machine can calculate \( \phi(x) \) in time polynomial in terms of the size of \( x \).

A problem \( A \) is NP-hard if, given any NP problem \( B \), there is a polynomial time reduction from \( B \) to \( A \). The usual way to show that a problem is NP-hard is to demonstrate a polynomial time reduction from another problem already known to be NP-hard. This is sufficient as the composition of two polynomial time reductions is also a polynomial time reduction.

NP-hardness is important because if a problem is both P and NP-hard then all NP problems are P, and thus the sets of P problems and NP problems are equal. It is famously unknown whether or not these sets are equal, so NP-completeness is the closest we can (currently) get to saying that a problem is NP but not P.
Let $\Omega$ be an identifier for a class of monoids ($B$, $R$, $UT$, $U$, $G$, $\bar{G}$, $C$, or $DC$). We define the following three decision problems on $\Omega_n$:

**Definition 1.14: $\mathcal{L}$-Order Problem ($\Omega LP$)**

Given $X,Y \in \Omega_n$, the $\mathcal{L}$-order problem asks if $X \leq_{\mathcal{L}} Y$.

**Definition 1.15: $\mathcal{R}$-Order Problem ($\Omega RP$)**

Given $X,Y \in \Omega_n$, the $\mathcal{R}$-order problem asks if $X \leq_{\mathcal{R}} Y$.

**Definition 1.16: $\mathcal{J}$-Order Problem ($\Omega JP$)**

Given $X,Y \in \Omega_n$, the $\mathcal{J}$-order problem asks if $X \leq_{\mathcal{J}} Y$.

In some of the classes of monoids we will also consider the following decision problem on $\Omega_n$:

**Definition 1.17: Membership Problem ($\Omega MP$)**

Given $C \in B_n$, the membership problem asks if $C \in \Omega_n$.

In each problem the size of the input is $n$, the order of the elements of $\Omega_n$. For each $\Omega$ and each $n$ the monoid $\Omega_n$ is finite, so there is a fixed bound on the amount of time required to solve each problem on inputs from $\Omega_n$. For each class of monoids we will find the computational complexity of each of the first three problems (with the exception of the double Catalan monoids where we omit the $\mathcal{J}$-order problem), and in some cases also the membership problem, by looking at how the upper bound on the amount of time needed to solve the problem varies with $n$. 
The following table summarises the main results of this thesis by categorising each of the considered problems as P or NP-complete (NPC):

<table>
<thead>
<tr>
<th>$\Omega$</th>
<th>$\Omega_{LP}$</th>
<th>$\Omega_{RP}$</th>
<th>$\Omega_{JP}$</th>
<th>$\Omega_{MP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{B}$</td>
<td>P</td>
<td>P</td>
<td>NPC</td>
<td>-</td>
</tr>
<tr>
<td>$R$</td>
<td>P</td>
<td>P</td>
<td>NPC</td>
<td>-</td>
</tr>
<tr>
<td>$UT$</td>
<td>P</td>
<td>P</td>
<td>NPC</td>
<td>-</td>
</tr>
<tr>
<td>$U$</td>
<td>P</td>
<td>P</td>
<td>NPC</td>
<td>-</td>
</tr>
<tr>
<td>$G$</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
</tr>
<tr>
<td>$\bar{G}$</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
</tr>
<tr>
<td>$C$</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>$DC$</td>
<td>P</td>
<td>P</td>
<td>-</td>
<td>P</td>
</tr>
</tbody>
</table>

We will show that some of these decision problems are NP-hard using polynomial time reductions from the dominating set problem and the 3-satisfiability problem, both of which are already known to be NP-complete (see [13, p. 75] and [21]).

**Definition 1.18: Dominating Set**

Let $G = (V, E)$ be a graph (a vertex set $V$ and an edge set $E$ which is a set of two-element subsets of $V$). We say that $D \subseteq V$ is a dominating set for $G$ if every vertex not in $D$ is connected to an element of $D$ via an edge:

$$v \in V \setminus D \implies \exists d \in D, \{v, d\} \in E.$$  

The dominating set problem is usually stated as follows: given a graph $G = (V, E)$ and a natural number $k \in \mathbb{N}$, does $G$ have a dominating set of size less than or equal to $k$?

We will instead use the following formulation which is clearly computationally equivalent and will allow us to ignore some edge cases:
1.3. STRUCTURE OF THE THESIS

Definition 1.19: Dominating Set Problem (DSP)
Given a graph \( G = (V, E) \) with vertex set \( V = \{1, \ldots, n\} \) and a natural number \( 0 < k < n \), the dominating set problem is the decision problem of determining whether or not \( G \) has a dominating set of size equal to \( k \).

Definition 1.20: 3-Satisfiability (3SAT)
Given a logical statement \( \psi \) in conjunctive normal form such that each conjunct is a disjunction of exactly three literals, the 3-satisfiability problem asks if there is a valuation \( v \) such that \( v(\psi) = 1 \).

1.3 Structure of the Thesis

Each of the previously defined classes of monoids will have its own chapter. In each chapter we will give some background for the monoids, including previous work and examples, then present some new results. In each case these new results will include proofs of the complexity classes of each of the three decision problems relating to Green’s pre-orders, with the exception of the double Catalan monoids, for which the \( J \)-order problem is omitted, and the Boolean monoids where these results are not new but we provide alternative proofs. Most chapters will also include other results about the relevant class of monoids.
Chapter 2

Boolean Monoids

The first class of monoids we look at are the monoids of all $n \times n$ Boolean matrices.

A Boolean algebra is defined (see for example [22, p. 1]) as a non-empty set $B$ with two binary operations $+$ and $\cdot$ such that:

1) both operations are commutative,
2) each operation distributes over the other,
3) the operations $+$ and $\cdot$ have identity elements 0 and 1 respectively,
4) each element $x$ has a complement $x^c$ such that $x + x^c = 1$ and $x \cdot x^c = 0$.

The name Boolean is used because Boolean logic, where 0 represents false and 1 represents true, provides an example of a two element Boolean algebra where $+$ represents or and $\cdot$ represents and. We refer to this two element Boolean algebra as the Boolean semiring. The operations on this semiring are the usual addition and multiplication except that $1 + 1 = 1$.

Definition 2.1

Let $\mathbb{B}$ be the set $\{0, 1\}$ equipped with the operations of addition and multiplication, with $1 + 1 = 1$:

$$\mathbb{B} = (\{0, 1\}, +, \cdot).$$
Let $\mathbb{B}_n$ be the set of all $n \times n$ matrices over $\mathbb{B}$. This set forms a monoid under matrix multiplication. We call this the Boolean monoid of rank $n$.

Right multiplying a matrix $A \in \mathbb{B}_n$ by a matrix $B \in \mathbb{B}_n$ has the effect of simultaneously, for each $j \leq n$, replacing column $j$ of $A$ with the (Boolean) sum of all columns $i$ of $A$ such that $b_{i,j} = 1$. Left multiplying $A$ by $B$ has the effect of simultaneously, for each $i \leq n$, replacing row $i$ of $A$ with the (Boolean) sum of all rows $j$ of $A$ such that $b_{i,j} = 1$.

Example 2.2
When giving examples of Boolean matrices we shall use a smaller font for the 0s than for the 1s to make the matrix structure easily visible. If we let $A, B \in \mathbb{B}_5$ be as follows

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

then the product $AB$ is equal to

$$AB = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$ 

Note that, for example, column 2 of $AB$ is equal to the sum of columns 2 and 4 of $A$ since column 2 of $B$ has 1s in positions 2 and 4.

These monoids, which are sometimes referred to as monoids of $(0,1)$-matrices, have been studied extensively. For example, Chaudhuri and Mukherjea have
described a process for constructing certain elements of \( B_n \), the details of which are contained in [5], such that the idempotents of \( B_n \) are precisely those matrices which can be constructed through that process.

In [24], Konieczny shows that a minimal generating set for \( B_n \) can be formed by taking a single representative of each \( D \)-class containing a prime element, along with four particular matrices, \( \delta \), \( \rho \), \( \pi \), and \( \tau \), which are defined in the paper. In the same paper Konieczny proves several other results about prime Boolean matrices.

\( B_n \) is isomorphic to the monoid of all binary relations on an \( n \) element set \( X \). Given such a relation \( R \), and an ordering of the elements of \( X \) as \( x_1, x_2, \ldots, x_n \), we can identify \( R \) with the matrix \( A \in B_n \) defined by

\[
a_{i,j} = \begin{cases} 
1 & \text{if } x_i R x_j, \\
0 & \text{otherwise}.
\end{cases}
\]

The monoid of all binary relations on \( X \) is often known as the full relation monoid on \( X \) and denoted \( B_X \). Multiplication of \( R, S \in B_X \) is

\[
RS = \{(a, b) : (a, c) \in R \text{ and } (c, b) \in S \text{ for some } c \in X\},
\]

as defined in [29]. Binary relations on an \( n \)-element set and the equivalent Boolean matrices are looked at in depth by Schwarz in [30], who also looks at some specific submonoids including the reflexive monoids covered in Chapter 3 of this thesis.

**Example 2.3**

The relation “\( i \) divides \( j \) or \( i + j = 8 \)” on the set \( \{1, 2, 3, 4, 5, 6\} \) corresponds to the matrix
Boolean matrices can also be interpreted as directed graphs. We identify a matrix \( A \in \mathbb{B}_n \) with the graph with vertex set \( \{1, 2, \ldots, n\} \) and edge set \( \{ (i, j) : a_{i,j} = 1 \} \). Under this interpretation, the product \( AB \) is the graph with edge set \( \{ (i, j) : (i, k) \) is an edge of \( A \) and \( (k, j) \) is an edge of \( B \) for some \( k \leq n \}. \) If \( G \) is the graph corresponding to the matrix \( A \) then \( A^p \) has a 1 in the \( (i, j) \) position if and only if there is a path of length \( p \) from \( i \) to \( j \) in \( G \).

**Example 2.3 (continued)**

The 6 × 6 matrix which corresponds to the relation “\( i \) divides \( j \) or \( i + j = 8 \)” also corresponds to the directed graph

![Diagram](image)

We define a partial order on \( \mathbb{B}_n \) as follows:

\[
A \preceq B \iff (a_{i,j} \leq b_{i,j} \text{ for all } i, j),
\]
and we define \( A \preceq B \) as "\( A \preceq B \) but not \( A = B \)". This partial order will be used throughout the thesis. Given Boolean matrices \( A, B \) and \( T \) such that \( A \preceq B \), we have

\[
(AT)_{i,j} = \sum_{k=1}^{n} a_{i,k} t_{k,j} \\
\leq \sum_{k=1}^{n} b_{i,k} t_{k,j} \\
= (BT)_{i,j}
\]

and so \( AT \preceq BT \). This shows that right multiplication by \( T \) is monotonic with respect to \( \preceq \). Similarly, left multiplication by \( T \) is also monotonic with respect to \( \preceq \), as in \( A \preceq B \implies TA \preceq TB \).

Several authors have considered Green’s relations on the monoid of Boolean matrices or other equivalent monoids. For example Plemmons and West [29] show that \( A =_L B \) if and only if \( A \) and \( B \) have the same row space, and \( A =_R B \) if and only if \( A \) and \( B \) have the same column space. In particular, the computational complexity of the related decision problems has been shown in [25] to be P for the \( L \) and \( R \)-order problems, and NP-complete for the \( J \)-order problem. We provide alternative proofs of these facts in Theorems 2.5 and 2.8, although the proof given in this thesis of the NP-hardness of the \( J \)-order problem uses a polynomial time reduction via another decision problem which does not appear in [25].

Before proving the computational complexity of these problems we prove a lemma which will be used throughout the thesis.

**Lemma 2.4**

Let \( M \subseteq \mathbb{B}_n \) be a monoid whose elements are \( n \times n \) Boolean matrices, and let \( M^T \) be the monoid \( \{X^T : X \in M\} \). For any \( X, Y \in M \), and considering the \( L \) and \( R \)-orders in \( M \) and \( M^T \) respectively, we have \( X \leq_L Y \iff X^T \leq_R Y^T \).
Proof. If \( X \leq_L Y \) then there exists \( U \in M \) such that \( X = UY \). Then \( X^T = Y^T U^T \) so there exists \( V \in M^T \) such that \( X^T = Y^T V \) and so \( X^T \leq_R Y^T \). The converse is analogous. \( \square \)

**Theorem 2.5**

The decision problems \( \mathbb{B}LP \) and \( \mathbb{B}RP \) are in \( P \).

Proof. Given \( X, Y \in \mathbb{B}_n \), \( \mathbb{B}LP \) asks if \( X \leq_L Y \) and \( \mathbb{B}RP \) asks if \( X \leq_R Y \). Since \( X \leq_L Y \iff X^T \leq_R Y^T \) and \( \mathbb{B}_n^T = \mathbb{B}_n \), it suffices to show that \( \mathbb{B}RP \) is in \( P \).

We define \( T \in \mathbb{B}_n \) as follows:

\[
t_{i,j} = \begin{cases} 
1 & \text{if } y_{k,i} \leq x_{k,j} \text{ for all } k \leq n, \\
0 & \text{otherwise.}
\end{cases}
\]

It can be checked in polynomial time if \( YT = X \), and we will show that this is true if and only if \( X \leq_R Y \). If \( YT = X \) holds then \( X \leq_R Y \) by definition of the \( \mathcal{R} \)-order. It remains to show that if \( X \leq_R Y \) then \( YT = X \).

Assume that \( V \in \mathbb{B}_n \) is such that \( YV = X \). We will show that \( X = YV \preceq YT \preceq X \), so that \( YT = X \). For the first inequality we consider arbitrary \( i, j \) such that \( v_{i,j} = 1 \). For each \( k \leq n \), we have \( x_{k,j} = \sum_{p=1}^{n} y_{k,p} v_{p,j} \geq y_{k,i} v_{i,j} = y_{k,i} \). Therefore \( y_{k,j} \leq x_{k,j} \) for all \( k \leq n \), and so \( t_{i,j} = 1 \) by definition. It follows that \( V \preceq T \), and so \( YV \preceq YT \) by monotonicity.

For the second inequality we consider arbitrary \( i, j \) such that \( (YT)_{i,j} = 1 \). This means there is some \( p \leq n \) such that \( y_{i,p} = 1 \) and \( t_{p,j} = 1 \). By the definition of \( T \) this means \( y_{k,p} \leq x_{k,j} \) for all \( k \leq n \), and therefore \( x_{i,j} = 1 \). Thus \( YT \preceq X \) as required. \( \square \)

For the sake of the \( \mathcal{J} \)-order problem we define another decision problem which will function as an intermediate step in a polynomial time reduction from DSP to \( \mathbb{B}JP \):
Definition 2.6: Boolean 2-Sided Transformation Problem (\(\mathbb{B}2TP\))

Given \(A, B \in \mathbb{B}^{m \times n}\), the Boolean 2-sided transformation problem asks if there exist \(R \in \mathbb{B}_m\) and \(S \in \mathbb{B}_n\) such that \(RAS = B\).

Theorem 2.7

The decision problem \(\mathbb{B}2TP\) is NP-hard.

Proof. Given \(A, B \in \mathbb{B}^{m \times n}\), \(\mathbb{B}2TP\) asks if there exist \(R \in \mathbb{B}_m\) and \(S \in \mathbb{B}_n\) such that \(RAS = B\). We show \(\mathbb{B}2TP\) is NP-hard via a polynomial time reduction from DSP.

Let \(H = (V, E)\) be a graph with vertex set \(V = \{1, \ldots, n\}\), and let \(k \in \mathbb{N}\) be such that \(0 < k < n\). We will construct matrices \(A, B \in \mathbb{B}^{(2n+2) \times (3n+2)}\) such that \(RAS = B\) for some \(R \in \mathbb{B}_{2n+2}\) and \(S \in \mathbb{B}_{3n+2}\) if and only if \(H\) has a dominating set of size \(k\).

We will assume that \(H\) has no isolated vertices. (If it does have any isolated vertices we can create a new graph \(G\) by removing one isolated vertex, and then \(H\) has a dominating set of size \(k\) if and only if \(G\) has a dominating set of size \(k - 1\). By successively removing vertices in this way we will either remove \(k\) vertices, in which case \(H\) does not have a dominating set of size \(k\), or we end up with an equivalent smaller instance of DSP where the graph has no isolated vertices.)

Recall that we identify \(n \times n\) Boolean matrices with directed graphs on the vertex set \(\{1, 2, \ldots, n\}\). We can regard the undirected graph \(H\) as a directed graph by replacing each undirected edge between \(i\) and \(j\) with directed edges from \(i\) to \(j\) and from \(j\) to \(i\). Then when we interpret \(H\) as a matrix it is the symmetric \(n \times n\) matrix

\[
h_{i,j} = \begin{cases} 
1 & \text{if } \{i, j\} \in E, \\
0 & \text{otherwise}.
\end{cases}
\]

Let \(A\) be the \((2n + 2) \times (3n + 2)\) Boolean matrix
\[
A = \begin{bmatrix}
H + I_n & I_{2n} & 0 & 0 \\
I_n & I_{2n} & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 \\
\end{bmatrix}
\]

where, in this matrix and all other matrices in this thesis, 0 and 1 represent submatrices consisting entirely of 0 and 1 respectively. Let \( B \) be the \((2n+2) \times (3n+2)\) Boolean matrix

\[
B = \begin{bmatrix}
1 & n-1 & 2n & 1 & 1 \\
1 & 0 & I_{2n} & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

If \( H \) has a dominating set \( D \) of size \( k \) then let \( v \in \mathbb{B}^n \) be defined by

\[
v_i = \begin{cases} 
1 & \text{if } i \in D, \\
0 & \text{otherwise}, 
\end{cases}
\]

and note that \( v \) has \( k \) 1s. By regarding \( H \) as a matrix we can take the product \( Hv \), which has a 1 in the \( i \)th position if and only if \( \{i, d\} \in E \) for some \( d \in D \). Thus \( Hv \) has a 1 in the \( i \)th position for each \( i \in V \setminus D \), and so \((H + I_n)v = Hv + v\) is the column vector of all 1s (which we shall write simply as \( 1 \)).

Conversely, if there exists a vector \( v \in \mathbb{B}^n \) with \( k \) 1s such that \((H + I_n)v = 1\), then for each \( i \leq n \) either \( I_nv \) has a 1 in position \( i \), in which case \( v_i = 1 \), or \( Hv \)
has a 1 in position $i$, in which case $\{i, d\} \in E$ for some $d$ such that $v_d = 1$. Thus the set $D = \{d : v_d = 1\}$ is a dominating set for $H$ of size $k$.

Therefore $H$ has a dominating set of size $k$ if and only if there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $(H + I_n)v = 1$.

Assume for now that there is such a vector and we will show that there exist $R \in \mathbb{B}_{2n+2}$ and $S \in \mathbb{B}_{3n+2}$ such that $RAS = B$. Since $v$ has $k$ 1s there is some $n \times n$ permutation matrix $P$ such that $Pv$ is the $n \times 1$ column vector with 1 in the first $k$ rows and 0 in the last $n - k$ rows.

Let $R$ be the $(2n + 2) \times (2n + 2)$ Boolean matrix

\[
R = \begin{bmatrix}
I_n & 0 & 0 & 0 \\
0 & P & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

and $S$ be the $(3n + 2) \times (3n + 2)$ Boolean matrix

\[
S = \begin{bmatrix}
v & 0 & 0 & 0 & 0 & 0 \\
0 & I_n & 0 & 0 & 0 \\
0 & 0 & P^{-1} & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Multiplying $R$ and $A$ gives
and multiplying this on the right by $S$ gives $B$. Thus there are $R \in \mathbb{B}_{2n+2}$ and $S \in \mathbb{B}_{3n+2}$ such that $RAS = B$.

Now conversely assume there exist $R \in \mathbb{B}_{2n+2}$ and $S \in \mathbb{B}_{3n+2}$ such that $RAS = B$ and we will show that $H$ has a dominating set of size $k$.

Since $B = RAS$, each row of $B$ is a linear combination of rows of $AS$. Row $2n + 1$ of $B$ has a 1 only in column $3n + 1$, so there must be a row of $AS$ which has a 1 in just column $3n + 1$. Each of the first $2n$ rows of $B$ has a 0 in column $3n + 2$, and exactly one 1 in one of the columns $n + 1, \ldots, 3n$, and each of these rows has its 1 in a distinct column. There must therefore be $2n$ rows of $AS$ each with a 1 in a distinct one of these columns and a 0 in column $3n + 2$. The last row of $B$ has a 1 in column $3n + 2$ so there must be a row of $AS$ with a 1 in this column, and it cannot be any of the previously mentioned rows as they all have a 0 in column $3n + 2$. There are only $2n + 2$ rows in $AS$ so the rows of $AS$ are as follows, although the order is currently unknown:

- $a)$ one row with a 1 in column $3n + 1$ and 0 in every other column,

- $b)$ $2n$ rows with a 1 in one of the columns $n + 1, \ldots, 3n$ (each in a different column) and a 0 in column $3n + 2$,

- $c)$ one row with a 1 in column $3n + 2$. 
At this point we note that for any $i$ and $j$, if $s_{i,j} = 1$ then column $j$ of $AS$ has a 1 in each row that column $i$ of $A$ does, since column $j$ of $AS$ is the sum of all columns $i$ of $A$ such that $s_{i,j} = 1$. Thus if column $j$ of $AS$ has fewer 1s than column $i$ of $A$ then $s_{i,j} = 0$. We can use this fact to determine some of the structure of $S$.

We shall label the blocks of $S$ as follows:

\[
S = \begin{bmatrix}
  \ & n & 2n & 1 & 1 \\
  S_{1,1} & S_{1,2} & S_{1,3} & S_{1,4} \\
  S_{2,1} & S_{2,2} & S_{2,3} & S_{2,4} \\
  S_{3,1} & S_{3,2} & S_{3,3} & S_{3,4} \\
  S_{4,1} & S_{4,2} & S_{4,3} & S_{4,4}
\end{bmatrix}
\]

Since we are assuming that $H$ has no isolated vertices, the matrix $H + I_n$ has at least two 1s in each column, and so each of the first $n$ columns of $A$ has at least three 1s. Column $3n + 1$ of $A$ also has at least three 1s, and the columns $n + 1, \ldots, 3n$ of $A$ each contain exactly two 1s.

From (a), (b) and (c) we have enough information about the rows of $AS$ to know that columns $n + 1, \ldots, 3n$ and $3n + 2$ each have a maximum of two 1s, and the last column has only one 1.

By the above argument we now know that blocks $S_{1,2}, S_{1,4}, S_{2,2}, S_{3,2}$, and $S_{3,4}$ are all equal to 0:
Multiplying on the left by $A$ gives us, for some matrices $\hat{S}_{i,j},$

\[
\hat{S} = \begin{pmatrix}
\hat{S}_{1,1} & 0 & \hat{S}_{1,3} & 0 \\
0 & \hat{S}_{2,2} & \hat{S}_{2,3} & 0 \\
0 & \hat{S}_{3,3} & 0 & \hat{S}_{3,4} \\
\hat{S}_{4,1} & \hat{S}_{4,2} & \hat{S}_{4,3} & \hat{S}_{4,4}
\end{pmatrix}
\]

The only row of $AS$ which can be of type (c) is row $2n + 2$, and so it must be of this type. Row $2n + 1$ can only possibly be of type (a). This leaves only the first $2n$ rows, which must therefore all be of type (b). Since each row of type (b) contains a single 1 in one of the columns $n+1, \ldots, 3n$, and each of these 1s is in a different column, the $2n \times 2n$ submatrix of $AS$ consisting of the intersection of rows $1, \ldots, 2n$ and columns $n+1, \ldots, 3n$ must be a permutation matrix.

From what we have deduced so far we know that, for some $2n \times 2n$ permutation matrix $P$, the matrix $AS$ is of the following form:

\[
AS = \begin{pmatrix}
\hat{S}_{1,1} & \hat{S}_{1,2} & \hat{S}_{0,3} & 0 \\
\hat{S}_{2,1} & 0 & \hat{S}_{2,3} & 0 \\
\hat{S}_{3,1} & \hat{S}_{3,2} & \hat{S}_{3,3} & \hat{S}_{3,4}
\end{pmatrix}
\]
CHAPTER 2. BOOLEAN MONOIDS

\[
AS = \begin{bmatrix}
\hat{S}_{1,1} & \hat{S}_{0,3} & 1 & 0 \\
0 & 0 & 1 & 0 \\
\hat{S}_{3,1} & \hat{S}_{3,2} & \hat{S}_{3,3} & 1
\end{bmatrix}
\]

Since row \(2n + 1\) of \(A\) contains a 1 in column \(3n + 1\) only, row \(2n + 1\) of \(AS\) is the same as row \(3n + 1\) of \(S\), and so row \(3n + 1\) of \(S\) is also of type (a). We therefore know that \(S_{3,1} = 0\) and \(S_{3,3} = 1\):

\[
S = \begin{bmatrix}
S_{1,1} & 0 & S_{1,3} & 0 \\
S_{2,1} & S_{2,2} & S_{2,3} & 0 \\
0 & 0 & 1 & 0 \\
\hat{S}_{4,1} & \hat{S}_{4,2} & \hat{S}_{4,3} & \hat{S}_{4,4}
\end{bmatrix}
\]

Multiplying on the left by \(A\) tells us that \(\hat{S}_{1,3}\) and \(\hat{S}_{3,3}\) are both equal to 1:
Since $RAS = B$ we can determine some of the structure of $R$ from what we know about $AS$ and $B$. We shall label the blocks of $R$ as follows:

\[
R = \begin{bmatrix}
\hat{S}_{1,1} & P^{-1} & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots \\
\hat{S}_{3,1} & S_{3,2} & 1 & 1 & \cdots \\
\end{bmatrix}
\]

The first $n$ rows of column $3n+1$ of $B$ are equal to 0. Thus for each 1 in column $3n+1$ of $AS$ there must be corresponding 0s in each of the first $n$ rows of $R$. Thus blocks $R_{1,2}$, $R_{1,3}$ and $R_{1,4}$ are all equal to 0. Rows $n+1, \ldots, 2n$ of column $3n+2$ of $B$ are equal to 0. Thus for each 1 in column $3n+2$ of $AS$ there must be corresponding 0s in each of the rows $n+1, \ldots, 2n$ of $R$. Thus block $R_{2,4}$ is equal to 0. Row $2n+2$ of $B$ has a 1 in column $3n+2$. Thus the only 1 in column $3n+2$ of $AS$ must correspond to a 1 in row $2n+2$ of $R$. Thus block $R_{4,4}$ is equal to 1.

The $2n \times 2n$ submatrix of $B$ consisting of the intersection of rows $1, \ldots, 2n$
with columns \( n + 1, \ldots, 3n \) is equal to \( I_{2n} \). When written as a product of blocks of \( R \) and \( AS \) this submatrix is
\[
I_{2n} = \begin{bmatrix}
R_{1,1} & 0 \\
\cdots & \cdots \\
R_{2,1} & R_{2,2}
\end{bmatrix}^{-1} + \begin{bmatrix}
0 \\
\cdots \\
R_{2,3}
\end{bmatrix}0 + \begin{bmatrix}
0 \\
\cdots \\
0
\end{bmatrix} \hat{S}_{3,2}.
\]

Thus the submatrix of \( R \) consisting of blocks \( R_{1,1}, R_{1,2}, R_{2,1} \) and \( R_{2,2} \) must be equal to \( P \). Since we already know that \( R_{1,2} = 0 \) we must have \( R_{1,1} = P_1, R_{2,2} = P_2 \) and \( R_{2,1} = 0 \) for some \( n \times n \) permutation matrices \( P_1 \) and \( P_2 \).

We now know that \( R \) is as follows:
\[
R = \begin{bmatrix}
\cdots & \cdots & \cdots & \cdots \\
P_1 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots \\
0 & P_2 & R_{2,3} & 0 \\
\cdots & \cdots & \cdots & \cdots \\
R_{3,1} & R_{3,2} & R_{3,3} & R_{3,4} \\
\cdots & \cdots & \cdots & \cdots \\
R_{4,1} & R_{4,2} & R_{4,3} & 1
\end{bmatrix}.
\]

Multiplying on the right by \( A \) gives us, for some matrices \( \hat{R}_{i,j} \),
\[
RA = \begin{bmatrix}
\cdots & \cdots & \cdots & \cdots \\
P_1 (H + I_n) & P_1 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots \\
P_2 & 0 & P_2 & 1 \\
\cdots & \cdots & \cdots & \cdots \\
\hat{R}_{3,1} & \hat{R}_{3,2} & \hat{R}_{3,3} & R_{3,4} R_{3,5} \\
\cdots & \cdots & \cdots & \cdots \\
\hat{R}_{4,1} & 1 & 1 & 1
\end{bmatrix}.
\]
Now, since $RAS = B$, we can look at $RA$ and $B$ to give us some new information about $S$.

In each of the first $n$ columns of $B$ there is a 0 in row $2n + 2$. Thus any 1 in row $2n + 2$ of $RA$ must correspond to a 0 in each of the first $n$ columns of $S$. Row $2n + 2$ of $RA$ has a 1 in each of the last $2n + 2$ columns, so we know that $S_{2,1}, S_{3,1}, S_{4,1}$ and $S_{5,1}$ are all equal to 0.

The submatrix of $B$ consisting of the intersection of rows $n + 1, \ldots, 2n$ with columns $2, \ldots, n$ is equal to 0. Thus any 1 in any of the rows $n + 1, \ldots, 2n$ of $RA$ corresponds to a 0 in each of the columns $2, \ldots, n$ of $S$. Between them, the rows $n + 1, \ldots, 2n$ of $RA$ contain a 1 in each of the first $n$ columns, so $S_{1,1}$ is equal to 0 in all columns except the first.

We now know that $S$ is of the following form for some $v \in \mathbb{B}^n$:

\[
S = \begin{bmatrix}
1 & n-1 & 2n & 1 & 1 \\
\vdots & 0 & 0 & S_{1,3} & 0 \\
0 & S_{2,2} & S_{2,3} & 0 & 2n \\
0 & 0 & 1 & 0 & 1 \\
0 & S_{4,2} & S_{4,3} & S_{4,4} & 1 \\
\end{bmatrix}
\]

Let $u \in \mathbb{B}^n$ be the vector in which the first $k$ rows are 1 and the rest are 0. The first column of rows $n + 1, \ldots, 2n$ of $B$ is equal to $u$, and by looking at the product of $RA$ and $S$ we see that $u = P_2 v$ and so $v = P_2^{-1} u$ and thus $v$ has $k$ entries equal to 1.
The first column of the first $n$ rows of $B$ is equal to 1, and by looking at the product of $RA$ and $S$ we see that $P_1(H + I_n)v = 1$ and so $(H + I_n)v = P_1^{-1}1 = 1$. Therefore $H$ has a dominating set of size $k$. □

**Theorem 2.8**

The decision problem $BJP$ is NP-complete.

*Proof.* Given $X, Y \in \mathbb{B}_n$, $BJP$ asks if $X \leq_J Y$, or in other words if there exist $U, V \in \mathbb{B}_n$ such that $X = UYV$.

An arbitrary element of $\mathbb{B}_n$ can be chosen by making $n^2$ choices of 0 or 1. Thus if there exist matrices $U, V \in \mathbb{B}_n$ such that $UYV = X$, a non-deterministic Turing machine can correctly guess them in polynomial time by simultaneously trying every possible combination of $U$ and $V$. This shows that $BJP$ is in NP. We show $BJP$ is NP-hard via a polynomial time reduction from $B^2TP$.

Let $A, B \in \mathbb{B}^{m \times n}$. If $m = n$ then $A$ and $B$ are already an instance of $BJP$. If $n < m$ then we note that $RAS = B \iff S^T A^T R^T = B^T$. Thus we can assume that $m < n$.

We will construct matrices $X, Y \in \mathbb{B}_n$ such that $UYV = X$ for some $U, V \in \mathbb{B}_n$ if and only if $RAS = B$ for some $R \in \mathbb{B}_m$ and $S \in \mathbb{B}_n$.

Let $X$ and $Y$ be the $n \times n$ Boolean matrices

$$X = \begin{bmatrix} \overbrace{B}^{n} & \cdots & \overbrace{0}^{n} \\ & \cdots & \\ \overbrace{0}^{n} & \cdots & \overbrace{0}^{n} \end{bmatrix}^{m}_{n - m}, \quad Y = \begin{bmatrix} \overbrace{A}^{n} & \cdots & \overbrace{0}^{n} \\ & \cdots & \\ \overbrace{0}^{n} & \cdots & \overbrace{0}^{n} \end{bmatrix}^{m}_{n - m}.$$  

First, assume there are $R \in \mathbb{B}_m$ and $S \in \mathbb{B}_n$ such that $RAS = B$. Let $U$ be the $n \times n$ matrix
and let $V = S$. Now $UYV = X$ as required.

Conversely, if there is $U, V \in \mathbb{B}_n$ such that $UYV = X$ then let $S = V$ and let $R$ be the top-left $m \times m$ submatrix of $U$, so that

$$U = \begin{bmatrix} m & n - m \\ R & 0 \\ 0 & 0 \end{bmatrix}^{m \times n-m}$$

Multiplying $U$ on the right by $Y$ and $V$ gives

$$UY = \begin{bmatrix} n \\ RA \\ U_{2,1}A \end{bmatrix}^{m \times n-m}, \quad UYV = \begin{bmatrix} n \\ RAS \\ U_{2,1}AS \end{bmatrix}^{m \times n-m}.$$

Since we know $UYV = X$ this means that $RAS = B$ as required. \qed
Chapter 3

Reflexive Monoids

Since we can identify Boolean matrices with binary relations we can interpret properties of binary relations such as reflexivity and symmetry in the context of Boolean matrices. The next class of monoids we look at consist of the reflexive Boolean matrices.

Definition 3.1: Reflexive Monoid

Let $R_n$ be the submonoid of $B_n$ consisting of all matrices which correspond to reflexive relations on an $n$ element set. These are precisely the matrices with only 1s on the diagonal. We call this the reflexive monoid of rank $n$.

In [27] Parker proves some basic properties of the reflexive monoids using an axiomatic approach. Schwarz proves a few more properties in the last section of [30], including that the idempotents of $R_n$ are precisely the $n \times n$ matrices corresponding to relations which are both reflexive and transitive. Their finite basis problem is considered by Volkov in [34]. The reflexive monoids seem to have otherwise not received much attention.

Since the multiplication is the same as in the Boolean monoids, right multiplying a matrix $A \in B_n$ by a matrix $B \in R_n$ still has the effect of simultaneously, for
each $j \leq n$, replacing column $j$ of $A$ with the sum of all columns $i$ of $A$ such that $b_{i,j} = 1$. In this case we have $b_{j,j} = 1$ for all $j$, so column $j$ of $A$ is always one of the columns which are summed to give column $j$ of $AB$, and so column $j$ of $A$ is always $\preceq$-below (or equal to) column $j$ of $AB$. The same observation holds for the rows of $A$ when left multiplying by $B$.

Monotonicity is very useful in $\mathbb{R}^n$ and submonoids of $\mathbb{R}^n$. If $T \in \mathbb{R}^n$ then we have $I_n \preceq T$ so for any Boolean matrix $A$ monotonicity gives us $A \preceq AT$ and $A \preceq TA$. We shall use monotonicity in this way throughout the thesis.

**Example 3.2**

Let $A \in \mathbb{B}_5$ and $B \in \mathbb{R}_5$ be as follows

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Then the product $AB$ is equal to

$$AB = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

The matrix $AB$ has a 1 in each position that $A$ does, so $A \preceq AB$.

**Theorem 3.3**

The decision problems RLP and RRP are in $P$.

**Proof.** Given $X, Y \in \mathbb{R}_n$, RLP asks if $X \preceq_L Y$ and RRP asks if $X \preceq_R Y$. Since $X \preceq_L Y \iff X^T \preceq_R Y^T$ and $\mathbb{R}_n^T = \mathbb{R}_n$ it suffices to show that RRP is in $P$. 

We define $T \in \mathbb{B}_n$ as in the proof of Theorem 2.5 by

$$t_{i,j} = \begin{cases} 1 & \text{if } y_{k,i} \leq x_{k,j} \text{ for all } k \leq n, \\ 0 & \text{otherwise}. \end{cases}$$

It can be checked in polynomial time if $T \in R_n$ by checking for 1s on the diagonal. It can also be checked in polynomial time if $YT = X$. We will show that these are true if and only if $X \leq_R Y$. Clearly, if $T \in R_n$ and $YT = X$ then $X \leq_R Y$ by the definition of the $R$-order. It remains to show that if $X \leq_R Y$ then $T \in R_n$ and $YT = X$.

Assume there exists some $V \in R_n$ such that $YV = X$. Then for each $k, i \leq n$ we have $v_{i,i} = 1$, so $x_{k,i} = \sum_{p=1}^{n} y_{k,p} v_{p,i} \geq y_{k,i} v_{i,i} = y_{k,i}$, and so $t_{i,i} = 1$ by definition. Thus $T \in R_n$. Since $V \in R_n \subseteq \mathbb{B}_n$ we have already shown, in the proof of Theorem 2.5, that $YT = X$. 

As with the Boolean monoids, we define another decision problem which will function as an intermediate step in a polynomial time reduction, in this case from 3SAT to RJP:

**Definition 3.4: Reflexive 2-Sided Transformation Problem (R2TP)**

Given $A, B \in \mathbb{B}_{m \times n}$, the reflexive 2-sided transformation problem asks if there exist $R \in R_m$ and $S \in R_n$ such that $RAS = B$.

**Theorem 3.5**

The decision problem R2TP is NP-hard.

**Proof.** Given $A, B \in \mathbb{B}_{m \times n}$, R2TP asks if there exist $R \in R_m$ and $S \in R_n$ such that $RAS = B$. We show R2TP is NP-hard via a polynomial time reduction from 3SAT.

Let $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} p_{i,j}$ be an instance of 3SAT with variables among $x_1, \ldots, x_t$. We will construct matrices $A, B \in \mathbb{B}_{(n+3t+1) \times (4n+t+1)}$ such that $RAS = B$ for some $R \in R_{n+3t+1}$ and $S \in R_{4n+t+1}$ if and only if $\psi$ is satisfiable.
Define, for each $i \leq n$, matrices $X(i), \neg X(i) \in \mathbb{B}^{t \times 3}$ and $E(i) \in \mathbb{B}^{n \times 3}$ by

$$x(i)_{k,j} = \begin{cases} 1 & \text{if } p_{i,j} = x_k, \\ 0 & \text{otherwise}, \end{cases}$$

$$\neg x(i)_{k,j} = \begin{cases} 1 & \text{if } p_{i,j} = \neg x_k, \\ 0 & \text{otherwise}, \end{cases}$$

$$e(i)_{k,j} = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{otherwise}. \end{cases}$$

Now define $A, B \in \mathbb{B}^{(n+3t+1) \times (4n+t+1)}$ as follows:

$$A = \begin{bmatrix} X(1) & X(2) & \cdots & X(n) & 0 & I_t & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \neg X(1) & \neg X(2) & \cdots & \neg X(n) & 0 & I_t & 0 \\ E(1) & E(2) & \cdots & E(n) & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ \end{bmatrix},$$

$$B = \begin{bmatrix} X(1) & X(2) & \cdots & X(n) & 1 & I_t & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \neg X(1) & \neg X(2) & \cdots & \neg X(n) & 1 & I_t & 0 \\ E(1) & E(2) & \cdots & E(n) & I_n & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 & I_t & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ \end{bmatrix}.$$
We first assume that $\psi$ is satisfiable and show that there exist $R \in R_{n+3t+1}$ and $S \in R_{4n+t+1}$ such that $RAS = B$. Let $T \subseteq \{1, \ldots, n\}$ be such that $\psi$ is satisfied by the valuation
\[ v(x_i) = \begin{cases} 1 & \text{if } i \in T, \\ 0 & \text{otherwise.} \end{cases} \]
Define $D(1), D(2) \in \mathbb{B}_t$ as follows:
\[ d(1)_{i,j} = \begin{cases} 1 & \text{if } i = j \notin T, \\ 0 & \text{otherwise,} \end{cases} \quad d(2)_{i,j} = \begin{cases} 1 & \text{if } i = j \in T, \\ 0 & \text{otherwise.} \end{cases} \]
Now define $R \in R_{n+3t+1}$ as follows:
\[
R = \begin{bmatrix}
I_t & 0 & 0 & 0 & 1 \\
0 & I_t & 0 & 0 & 1 \\
0 & 0 & I_n & 0 & 1 \\
D(1) & D(2) & 0 & I_t & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Since $v(\psi) = 1$, each conjunct $\bigvee_{j=1}^3 p_{i,j}$ has at least one disjunct $p_{i,j}$ such that $v(p_{i,j}) = 1$. Let $f : \{1, \ldots, n\} \to \{1, 2, 3\}$ be a function which maps each $i \leq n$ to a $j \in \{1, 2, 3\}$ such that $v(p_{i,j}) = 1$.

Define, for each $k \leq n$, the matrix $C(k) \in \mathbb{B}^{3 \times n}$ as follows:
\[ c(k)_{i,j} = \begin{cases} 1 & \text{if } j = k \text{ and } i = f(j), \\ 0 & \text{otherwise.} \end{cases} \]
Now define $S \in R_{4n+t+1}$ as follows:
Since $R$ is a reflexive matrix we know that $A \preceq RA$, and the differences between $A$ and $RA$ are because of the off-diagonal 1s in $R$. The matrix $R$ has four non-zero off-diagonal blocks, so multiplying $A$ on the left by $R$ does the following four things:

- row $n + 3t + 1$ is copied onto each row from 1 to $t$,
- row $n + 3t + 1$ is copied onto each row from $t + 1$ to $2t$,
- for each $k \leq t$, if $k \notin T$ then row $k$ is copied onto row $n + 2t + k$,
- for each $k \leq t$, if $k \in T$ then row $t + k$ is copied onto row $n + 2t + k$.

For each $k \leq t$ we note that both $a_{k,4n+k} = 1$ and $a_{t+k,4n+k} = 1$, so whether $k \in T$ or not, we end up with $(RA)_{n+2t,k,4n+k} = 1$.

For each $k \leq t$, consider each $i,j$ such that $p_{i,j} = x_k$. For each such $i,j$ we have $x(i)_{k,j} = 1$. This means that for each such $i,j$ there is a 1 in row $k$ of $A$. If $v(p_{i,j}) = 0$ then $k \notin T$ and so these 1s are copied into row $n + 2t + k$. If $v(p_{i,j}) = 1$ then these 1s are not copied.
Now consider each \( i, j \) such that \( p_{i,j} = \neg x_k \). For each such \( i, j \) we have \( \neg x(i)_{k,j} = 1 \). This means that for each such \( i, j \) there is a 1 in row \( t + k \) of \( A \). If \( v(p_{i,j}) = 0 \) then \( k \in T \) and so these 1s are copied into row \( n + 2t + k \). If \( v(p_{i,j}) = 1 \) then these 1s are not copied. We see that in either case, the 1s are copied to row \( n + 2t + k \) if and only if \( v(p_{i,j}) = 0 \).

Define, for each \( k \leq n \), the matrix \( Y(k) \in \mathbb{B}^{t \times 3} \) as follows:

\[
y(k)_{i,j} = \begin{cases} 
1 & \text{if } v(p_{k,j}) = 0 \text{ and } p_{k,j} = x_i \text{ or } \neg x_i, \\
0 & \text{otherwise.}
\end{cases}
\]

Now we see that

\[
RA = \begin{bmatrix}
X(1) & X(2) & \cdots & X(n) & 1 & I_t & 0 \\
\neg X(1) & \neg X(2) & \cdots & \neg X(n) & 1 & I_t & 0 \\
E(1) & E(2) & \cdots & E(n) & 0 & 0 & 0 \\
Y(1) & Y(2) & \cdots & Y(n) & 0 & I_t & 1 \\
0 & 0 & \cdots & 0 & 1 & 0 & 0
\end{bmatrix}^{t \times 1}.
\]

The matrix \( S \) is also reflexive and has \( 2n \) non-zero off-diagonal blocks, so multiplying \( RA \) on the right by \( S \) does the following 2 things for each \( i \leq n \):

- column \( 4n + t + 1 \) is copied onto each column from \( 3(i - 1) + 1 \) to \( 3i \),
- column \( 3(i - 1) + f(i) \) is copied onto column \( 3n + i \).
For each $i, k \leq n$, note that $(RA)_{2t+k,3(i-1)+f(i)} = e(i)_{k,f(i)}$, which is equal to 1 if and only if $k = i$. Thus, independently of the function $f$, we get $(RAS)_{2t+i,3n+i} = 1$ for each $i \leq n$ and $(RAS)_{2t+j,3n+i} = 0$ for all $j \neq i$.

For each $i \leq n$ we also note that $v(p_i,f(i)) = 1$, and we therefore have $y(i)_{k,f(i)} = 0$ for all $k \leq t$. Thus, multiplying $RA$ on the right by $S$ does not copy any of the 1s from the submatrices of the form $Y(i)$.

We therefore have

$$RAS = \begin{bmatrix}
X(1) & X(2) & \cdots & X(n) & 1 & I_t & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\neg X(1) & \neg X(2) & \cdots & \neg X(n) & 1 & I_t & 0 \\
E(1) & E(2) & \cdots & E(n) & I_n & 0 & 0 \\
1 & 1 & \cdots & 1 & 0 & I_t & 1 \\
0 & 0 & \cdots & 0 & 1 & 0 & 0 \\
\end{bmatrix},$$

which is equal to $B$ as required.

Now we assume that there exist $R \in R_{n+3t+1}$ and $S \in R_{4n+t+1}$ such that $RAS = B$ and we show that $\psi$ is satisfiable. We consider the possible structure of $R$ and $S$ by considering the left action of $R$ on $A$ and the right action of $S$ on $RA$.

First we consider the fact that $B$ has a block equal to $I_n$ with top-left corner at $b_{2t+1,3n+1}$. This means that for each $i \leq n$ we have $b_{2t+i,3n+i} = 1$. This 1 cannot be placed in this position by the left action of $R$ on $A$, as the only 1 on this column
of $A$ is in row $n + 3t + 1$, and copying this row into row $2t + i$ would result in off-diagonal 1s in this block.

Thus, the right action of $S$ on $RA$ places a 1 in $b_{2t+i,3n+i}$. Other than this 1, row $2t + i$ is identical in matrices $A$ and $B$, and therefore by monotonicity this row is identical in matrices $A$ and $RA$. The only columns of $RA$ which contain a 1 in this row are therefore columns $3(i - 1) + 1, 3(i - 1) + 2, \text{ and } 3(i - 1) + 3$, so right multiplication by $S$ copies at least one of these columns into column $2t + i$.

For each $i \leq n$, right multiplication by $S$ copies at least one column $3(i - 1) + j$ with $j \in \{1, 2, 3\}$ to column $2t + i$. We can therefore define a function $f : \{1, \ldots, n\} \rightarrow \{1, 2, 3\}$ such that for each $i \leq n$, column $3(i - 1) + f(i)$ is copied to column $3n + i$.

We next consider the fact that $B$ has a block equal to $I_t$ with top-left corner at $b_{n+3t+1,4n+1}$. This means that for each $k \leq t$ we have $b_{n+2t+k,4n+k} = 1$. This 1 cannot be placed there by the right action of $S$ on $RA$, as this would either require copying one of the first $3n$ columns into column $4n + k$, which would give us $b_{2t+i,4n+k} = 1$ for some $i \leq n$, or require copying the last column into column $4n + k$, which would give us $b_{n+2t+i,4n+k} = 1$ for all $i \leq n$. Since $b_{2t+i,4n+k} = 0$ for all $i \leq n$ and $b_{n+2t+i,4n+k} = 0$ for all $i, k \leq n$ such that $i \neq k$, this is not the case.

Thus, for each $k \leq n$, the left action of $R$ on $A$ places a 1 in $b_{n+2t+k,4n+k}$. Column $4n + k$ of $A$ only has 1s in rows $k$ and $t + k$. Thus, left multiplication by $R$ copies either row $k$ or row $t + k$ into row $n + 2t + k$, so either $r_{n+2t+k,k} = 1$ or $r_{n+2t+k,t+k} = 1$. We can define $T = \{k \leq n : r_{n+2t+k,t+k} = 1\}$. We now know that for each $k \in \{1, \ldots, n\} \setminus T$, we have $r_{n+2t+k,k} = 1$.

We will now show that $\psi$ is satisfied by the valuation defined by

$$v(x_k) = \begin{cases} 1 & \text{if } k \in T, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, we will show that for each $i \leq n$ this valuation gives us $v(p_i,f(i)) =$
1. Assume for contradiction that \( v(p_{i,f(i)}) = 0 \). Either \( p_{i,f(i)} = x_k \) for some \( k \leq t \) or \( p_{i,f(i)} = \neg x_k \) for some \( k \leq t \).

In the first case we have \( v(x_k) = 0 \), so by the definition of \( v \) we have \( k \not\in T \).

Thus \( r_{n+2t+k,k} = 1 \). By the definition of submatrix \( X(i) \) we know that \( x(i)_{k,f(i)} = 1 \), and so \( a_{k,3(i-1)+f(i)} = 1 \). Therefore \( (RA)_{n+2t+k,3(i-1)+f(i)} = 1 \).

In the second case we have \( v(\neg x_k) = 0 \), so \( v(x_k) = 1 \). By the definition of \( v \) we have \( k \in T \). Thus \( r_{n+2t+k,t+k} = 1 \). By the definition of submatrix \( \neg X(i) \) we know that \( \neg x(i)_{k,f(i)} = 1 \), and so \( a_{t+k,3(i-1)+f(i)} = 1 \). Again, this gives us \( (RA)_{n+2t+k,3(i-1)+f(i)} = 1 \).

In either case we have \( (RA)_{n+2t+k,3(i-1)+f(i)} = 1 \) and, by the definition of \( f \), the right action of \( S \) on \( RA \) copies column \( 3(i-1) + f(i) \) to column \( 3n+i \). We therefore have \( (RAS)_{n+2t+k,3n+i} = 1 \). However, \( RAS = B \), and \( b_{n+2t+k,3n+i} = 0 \) for all \( k \) and \( i \), giving the required contradiction. \( \square \)

**Theorem 3.6**

The decision problem RJP is NP-complete.

*Proof.* Given \( X, Y \in R_n \), RJP asks if \( X \leq_J Y \), or in other words if there exist \( U, V \in R_n \) such that \( UYV = X \).

An element of \( R_n \) consists of \( n \) 1s on the diagonal and \( n(n-1) \) off-diagonal entries which are either 0 or 1. A non-deterministic Turing machine can therefore guess an element of \( R_n \) in polynomial time by making \( n(n-1) \) choices between 0 and 1, and can thus check in polynomial time whether or not there exist \( U, V \in R_n \) such that \( UYV = X \). This shows that RJP is in NP. We show RJP is NP-hard via a polynomial time reduction from R2TP.

Let \( A, B \in \mathbb{B}^{m \times n} \). We will construct matrices \( X, Y \in R_{m+n+1} \) such that \( UYV \neq X \) for some \( U, V \in R_{m+n+1} \) if and only if \( RAS = B \) for some \( R \in R_m \) and \( S \in R_n \).

Let \( X \) and \( Y \) be the \((m+n+1) \times (m+n+1)\) reflexive matrices
First we assume there are \( R \in R_m \) and \( S \in R_n \) such that \( RAS = B \) and we show that there are \( U, V \in R_{m+n+1} \) such that \( UYV = X \). Let \( U \) and \( V \) be the reflexive \((m+n+1) \times (m+n+1)\) matrices

\[
X = \begin{bmatrix}
1 & 0 & B \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
Y = \begin{bmatrix}
1 & 0 & A \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

It is simple to check that \( UYV = X \).

Now we assume that there are \( U, V \in R_{m+n+1} \) such that \( UYV = X \) and we show there are \( R \in R_m \) and \( S \in R_n \) such that \( RAS = B \). We consider the possible structure of \( U \) and \( V \).

If \( u_{i,j} = 1 \) then for each \( k \leq m+n+1 \) we have \( y_{j,k} \leq x_{i,k} \). In other words if for some \( k \) we have \( y_{j,k} = 1 \) and \( x_{i,k} = 0 \) then \( u_{i,j} = 0 \). We can tell from this that for each \( i \leq m \) and \( j > m \) we have \( u_{i,j} = 0 \), and so \( U \) is equal to

\[
U = \begin{bmatrix}
R & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & I_n \\
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
I_m & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & S \\
\end{bmatrix}
\]
for some \( R \in R_m \) and matrices \( U_{a,b} \) of the relevant sizes.

If \( v_{i,j} = 1 \) then for each \( k \leq m + n + 1 \) we have \( y_{k,i} \leq x_{k,j} \). In other words if for some \( k \) we have \( y_{k,i} = 1 \) and \( x_{k,j} = 0 \) then \( v_{i,j} = 0 \). We can tell from this that for each \( j > m + 1 \) and \( i \leq m + 1 \) we have \( v_{i,j} = 0 \), and so \( V \) is equal to

\[
V = \begin{bmatrix}
V_{1,1} & V_{1,2} & 0 \\
V_{2,1} & 1 & 0 \\
V_{3,1} & V_{3,2} & S
\end{bmatrix}
\]

for some \( S \in R_n \) and matrices \( V_{a,b} \) of the relevant sizes.

Multiplying \( UYV \) gives

\[
UYV = \begin{bmatrix}
1V_{1,1} + RAV_{3,1} & 1V_{1,2} + RAV_{3,2} & RAS \\
U_{2,3}AV_{3,1} + U_{2,3}1V_{3,1} & 1 & U_{2,3}AS + U_{2,3}1 \\
U_{3,1}1V_{1,1} + U_{3,2}1V_{1,1} + U_{3,1}1V_{1,2} + & U_{3,1}AS + U_{3,3}1 & U_{3,1}AS + U_{3,3}1 \\
U_{3,3}1V_{2,1} + U_{3,3}1V_{3,1} + U_{3,3}1 & & U_{3,3}1 + \\
U_{3,3}1V_{3,1} & U_{3,1}AV_{3,2} &
\end{bmatrix}
\]

Comparing the top-right blocks of \( UYV \) and \( X \) gives us \( RAS = B \) as required.
Whereas some of the monoids considered in this thesis are defined by a generating set, the reflexive monoids are defined by their structure. It would therefore be of interest to find a minimal generating set for $R_n$ for each $n$. We have used MATLAB to calculate such a minimal generating set for $n \leq 5$ in the hope of identifying some patterns. Unfortunately the code does not run in a reasonable amount of time for $n \geq 6$ and the generating sets for $n \leq 5$ reveal no obvious patterns.

Below we list some elements of $R_2$, $R_3$, $R_4$, and $R_5$ as directed graphs. For each $n \leq 5$ the set of elements which are graph isomorphic to a listed element forms a generating set for $R_n$. 

\[ n = 2: \]

\[ n = 3: \]

\[ n = 4: \]
$n = 5$: 

[Diagram of various graphs with 5 nodes, each node connected by directed edges in different configurations.]
Chapter 4

Upper Triangular Monoids

The next class of monoids we consider are the submonoids of $\mathbb{B}_n$ consisting of only upper triangular matrices.

**Definition 4.1: Upper Triangular Monoid**

Let $UT_n$ be the submonoid of $\mathbb{B}_n$ consisting of elements which satisfy

$$i > j \implies a_{i,j} = 0.$$

We call this the upper triangular monoid of rank $n$.

Again, right multiplying a matrix $A \in \mathbb{B}_n$ by a matrix $B \in UT_n$ has the effect of simultaneously, for each $j \leq n$, replacing column $j$ of $A$ with the sum of all columns $i$ of $A$ such that $b_{i,j} = 1$. In this case the columns $i$ of $A$ such that $b_{i,j} = 1$ all lie to the left of column $j$, possibly including column $j$ itself. Similarly, left multiplying $A$ by $B$ results in a matrix where row $i$ is the sum of some rows which lie below (or are equal to) row $i$ of $A$.

The monoid consisting of all lower triangular $n \times n$ Boolean matrices is clearly isomorphic to $UT_n$ by the map sending each matrix to its transpose. All results in
this chapter therefore have analogous results in the monoids of lower triangular Boolean matrices.

We can define a function \( U : \mathbb{B}_n \rightarrow UT_n \) by letting \( Y = U(X) \) be the above-diagonal part of \( X \):

\[
y_{i,j} = 1 \iff x_{i,j} = 1 \text{ and } i \leq j.
\]

Similarly we can define a map \( L \) to the monoid of lower triangular matrices by taking the below diagonal part of the matrix. These maps are not homomorphisms. For example if we let

\[
A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

then \( A^2 = I_2 \), so \( U(A^2) = I_2 \), but \( U(A)^2 \) is the zero matrix. We will make use of these functions in the chapter on the double Catalan monoids, where we will see that their restrictions to those monoids are homomorphisms.

**Theorem 4.2**

*The decision problems UTLP and UTRP are in P.*

*Proof.* Given \( X,Y \in UT_n \), UTLP asks if \( X \leq_L Y \) and UTRP asks if \( X \leq_R Y \).

Given a matrix \( M \) let \( M^R \) be result of rotating the matrix by 180 degrees. Then for matrices \( A, B \) and \( C \), we have \( AB = C \) if and only if \( A^R B^R = C^R \). For any \( A \in \mathbb{B}_n \) we see that \( A \in UT_n \) if and only if \( A^RT \in UT_n \). Since \( X = UY \iff X^{RT} = Y^{RT} U^{RT} \), we have \( X \leq_L Y \iff X^{RT} \leq_R Y^{RT} \), so the problems UTLP and UTRP have the same complexity class and it is enough to prove the result for just UTRP.

We define \( T \in \mathbb{B}_n \) as in the proof of Theorem 2.5 by

\[
t_{i,j} = \begin{cases} 
1 & \text{if } y_{k,i} \leq x_{k,j} \text{ for all } k \leq n, \\
0 & \text{otherwise}.
\end{cases}
\]
We have seen that $YT = X$ if and only if there exists $V \in \mathbb{B}$ such that $YV = X$. We claim that $YU(T) = X$ if and only if there exists $V \in UT_n$ such that $YV = X$. It is possible to check in polynomial time if $YU(T) = X$ so proving this claim is sufficient to prove the theorem.

Clearly if $YU(T) = X$ then there exists $V \in UT_n$ such that $YV = X$. It remains to show that if such a $V$ exists then $YU(T) = X$.

Assume that $V \in UT_n$ is such that $YV = X$. We will show that $X = YV \preceq YU(T) \preceq X$, so that $YU(T) = X$ as required. For the first inequality we consider arbitrary $i, j$ such that $v_{i,j} = 1$. Since $V \in UT_n$ we have $i \leq j$. For each $k \leq n$, we have $x_{k,j} = \sum_{p=1}^{n} y_{k,p} v_{p,j} \geq y_{k,i} v_{i,j} = y_{k,i}$. Therefore $y_{k,i} \leq x_{k,j}$ for all $k \leq n$, so $t_{i,j} = 1$ by definition, and $i \leq j$ so $U(T)_{i,j} = 1$. It follows that $V \preceq U(T)$, and so $YV \preceq YU(T)$.

For the second inequality we consider arbitrary $i, j$ such that $(YU(T))_{i,j} = 1$. This means there is some $p \leq n$ such that $y_{i,p} = 1$ and $U(T)_{p,j} = 1$. Since $U(T)_{p,j} = 1$ we also have $t_{p,j} = 1$ and by the definition of $T$ we have $y_{k,p} \leq x_{k,j}$ for all $k \leq n$. Therefore $x_{i,j} = 1$ and so $YU(T) \preceq X$ as required.

**Theorem 4.3**

The decision problem UTJP is NP-complete.

**Proof.** Given $X, Y \in UT_n$, UTJP asks if $X \preceq Y$, or in other words if there exist $U, V \in UT_n$ such that $UYV = X$.

A non-deterministic Turing machine can guess an element of $UT_n$ in polynomial time by selecting 0 or 1 for each of the $n(n+1)/2$ above diagonal elements. It can therefore check in polynomial time whether or not there exist $U, V \in UT_n$ such that $UYV = X$. This shows that UTJP is in NP. We show that UTJP is NP-hard via a polynomial time reduction from $\mathbb{B}$JP.
Let $X_0, Y_0 \in \mathbb{B}_n$. We will construct matrices $X, Y \in UT_{4n}$ such that $UYV = X$ for some $U, V \in UT_{4n}$ if and only if $U_0Y_0V_0 = X_0$ for some $U_0, V_0 \in \mathbb{B}_n$.

Let $X$ and $Y$ be the $4n \times 4n$ upper triangular matrices

$$X = \begin{bmatrix}
0 & 0 & 0 & X_0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \quad \quad Y = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & Y_0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}$$

First we assume there are $U_0, V_0 \in \mathbb{B}_n$ such that $U_0Y_0V_0 = X_0$ and we show that there are $U, V \in UT_{4n}$ such that $UYV = X$. Let $U$ and $V$ be the upper triangular $4n \times 4n$ matrices

$$U = \begin{bmatrix}
0 & U_0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \quad \quad V = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & V_0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}$$

It is easy to verify that $UYV = X$.

Now we assume that there are $U, V \in UT_{4n}$ such that $UYV = X$ and we show that there are $U_0, V_0 \in \mathbb{B}_n$ such that $U_0Y_0V_0 = X_0$. We write

$$U = \begin{bmatrix}
U_{1.1} & U_{1.2} & U_{1.3} & U_{1.4} \\
0 & U_{2.2} & U_{2.3} & U_{2.4} \\
0 & 0 & U_{3.3} & U_{3.4} \\
0 & 0 & 0 & U_{4.4} \\
\end{bmatrix} \quad \quad V = \begin{bmatrix}
V_{1.1} & V_{1.2} & V_{1.3} & V_{1.4} \\
0 & V_{2.2} & V_{2.3} & V_{2.4} \\
0 & 0 & V_{3.3} & V_{3.4} \\
0 & 0 & 0 & V_{4.4} \\
\end{bmatrix}$$

for some

\[ \begin{array}{c}
U_{1,2}, U_{1,3}, U_{1,4}, U_{2,3}, U_{2,4}, U_{3,4}, V_{1,2}, V_{1,3}, V_{1,4}, V_{2,3}, V_{2,4}, V_{3,4} \in \mathbb{B}_n,
\end{array} \] 

\[ U_{1,1}, U_{2,2}, U_{3,3}, U_{4,4}, V_{1,1}, V_{2,2}, V_{3,3}, V_{4,4} \in UT_n. \]

Then left multiplying \( Y \) by \( U \) gives

\[
UY = \begin{bmatrix}
\begin{array}{cccc}
0 & 0 & U_{1,2}Y_0 & 0 \\
0 & 0 & U_{2,2}Y_0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}
\end{bmatrix}
\]

and right multiplying \( UY \) by \( V \) gives

\[
UYV = \begin{bmatrix}
\begin{array}{cccc}
0 & 0 & U_{1,2}Y_0V_{3,3} & U_{1,2}Y_0V_{3,4} \\
0 & 0 & U_{2,2}Y_0V_{3,3} & U_{2,2}Y_0V_{3,4} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}
\end{bmatrix}
\].

Since \( UYV = X \), comparing the top right \( n \times n \) block of these matrices gives \( U_{1,2}Y_0V_{3,4} = X_0 \), so there exist \( U_0, V_0 \in \mathbb{B}_n \) such that \( U_0Y_0V_0 = X_0 \) as required. \( \square \)
Chapter 5

Unitriangular Monoids

If a matrix is both upper triangular and reflexive we call it unitriangular.

Definition 5.1: Unitriangular Monoid

Let \( U_n = R_n \cap UT_n \). We call this the unitriangular monoid of rank \( n \).

Again, right multiplying a matrix \( A \in B_n \) by a matrix \( B \in U_n \) has the effect of simultaneously, for each \( j \leq n \), replacing column \( j \) of \( A \) with the sum of all columns \( i \) of \( A \) such that \( b_{i,j} = 1 \). In this case the columns \( i \) of \( A \) such that \( b_{i,j} = 1 \) all lie to the left of column \( j \), and this always includes column \( j \) itself. Similarly, left multiplying \( A \) by \( B \) results in a matrix where row \( i \) is the sum of row \( i \) of \( A \) with some rows of \( A \) which lie below row \( i \).

The unitriangular monoids play an important role in the theory of \( J \)-trivial monoids: in [28] it is shown that a monoid \( M \) is \( J \)-trivial if and only if it divides \( U_n \) for some \( n \), where we say that \( M \) divides \( N \) if \( M \) is a quotient of some submonoid of \( N \) (see [11]).

In the same paper Pin and Straubing link the unitriangular monoids to the upper triangular monoids by showing that a monoid \( M \) divides \( UT_n \) for some \( n \).
if and only if it divides $\mathcal{P}(U_{n'})$ for some $n'$, where the multiplication on $\mathcal{P}(U_{n'})$ is defined as $XY = \{xy : x \in X \text{ and } y \in Y\}$.

**Theorem 5.2**

The decision problems ULP and URP are in $P$.

*Proof.* Given $X, Y \in U_n$, ULP asks if $X \leq_L Y$ and URP asks if $X \leq_R Y$.

Recall that $M^R$ is the result of rotating the matrix $M$ by 180 degrees. Since $X = UY \iff X^{RT} = Y^{RT} U^{RT}$ and $A \in U_n \iff A^{RT} \in U_n$, we have $X \leq_L Y \iff X^{RT} \leq_R Y^{RT}$, so the problems ULP and URP have the same complexity class and it is enough to prove the result for just URP.

We define $T \in \mathbb{B}$ as in the proof of Theorem 2.5 by

$$t_{i,j} = \begin{cases} 1 & \text{if } y_{k,i} \leq x_{k,j} \text{ for all } k \leq n, \\ 0 & \text{otherwise}. \end{cases}$$

Since $X, Y \in U_n = R_n \cap UT_n$ we know, from the proof of Theorem 3.3, that $T \in R_n$ and $YT = X$ if and only if there exists $V \in R_n$ such that $YV = X$ and, from the proof of Theorem 4.2, that $YU(T) = X$ if and only if there exists $V \in UT_n$ such that $YV = X$.

It can be checked in polynomial time if $T \in R_n$ and $YU(T) = X$. Clearly if these both hold then there exists $V \in U_n$ such that $YV = X$. Conversely if there exists $V \in U_n$ such that $YV = X$ then since $V \in R_n$ we have $T \in R_n$ and since $V \in UT_n$ we have $YU(T) = X$. 

**Theorem 5.3**

The decision problem UJP is NP-complete.

*Proof.* Given $X, Y \in U_n$, UJP asks if $X \leq_J Y$, or in other words if there exist $U, V \in U_n$ such that $UYV = X$. 

A non-deterministic Turing machine can guess an element of $U_n$ in polynomial time by selecting 0 or 1 for each of the $n(n - 1)/2$ elements strictly above the diagonal. It can therefore check in polynomial time whether or not there exist $U, V \in U_n$ such that $UYV = X$. This shows that UJP is in NP. We show that UJP is NP-hard via a polynomial time reduction from BJP.

Let $X_0, Y_0 \in \mathbb{B}_n$. We will construct matrices $X, Y \in U_{4n+2}$ such that $UYV = X$ for some $U, V \in U_{4n+2}$ if and only if $U_0Y_0V_0 = X_0$ for some $U_0, V_0 \in \mathbb{B}_n$.

Let $X$ and $Y$ be the $(4n + 2) \times (4n + 2)$ unitriangular matrices

$$X = \begin{bmatrix} 1 & n & n & n & n & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & I_n & 1 & 1 & X_0 & 0 \\ 0 & 0 & I_n & Y_0 & 1 & 1 \\ 0 & 0 & 0 & I_n & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$Y = \begin{bmatrix} 1 & n & n & n & n & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & I_n & 1 & 0 & 0 & 0 \\ 0 & 0 & I_n & Y_0 & 1 & 0 \\ 0 & 0 & 0 & I_n & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

First we assume there are $U_0, V_0 \in \mathbb{B}_n$ such that $U_0Y_0V_0 = X_0$ and we show that there are $U, V \in U_{4n+2}$ such that $UYV = X$. Let $U$ and $V$ be the unitriangular $(4n + 2) \times (4n + 2)$ matrices

$$U = \begin{bmatrix} 1 & n & n & n & n & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_n & U_0 & 0 & 0 & 0 \\ 0 & 0 & I_n & 0 & 1 & 0 \\ 0 & 0 & 0 & I_n & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$V = \begin{bmatrix} 1 & n & n & n & n & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_n & 0 & 1 & 0 & 0 \\ 0 & 0 & I_n & 0 & 0 & 0 \\ 0 & 0 & 0 & I_n & V_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
It is easy to verify that \( UYV = X \).

Now we assume that there are \( U, V \in U_{4n+2} \) such that \( UYV = X \) and we show that there are \( U_0, V_0 \in B_n \) such that \( U_0Y_0V_0 = X_0 \). We write

\[
U = \begin{bmatrix}
1 & U_{1,2} & U_{1,3} & U_{1,4} & U_{1,5} & U_{1,6} \\
0 & U_{2,2} & U_{2,3} & U_{2,4} & U_{2,5} & U_{2,6} \\
0 & 0 & U_{3,3} & U_{3,4} & U_{3,5} & U_{3,6} \\
0 & 0 & 0 & U_{4,4} & U_{4,5} & U_{4,6} \\
0 & 0 & 0 & 0 & U_{5,5} & U_{5,6} \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
1 & V_{1,2} & V_{1,3} & V_{1,4} & V_{1,5} & V_{1,6} \\
0 & V_{2,2} & V_{2,3} & V_{2,4} & V_{2,5} & V_{2,6} \\
0 & 0 & V_{3,3} & V_{3,4} & V_{3,5} & V_{3,6} \\
0 & 0 & 0 & V_{4,4} & V_{4,5} & V_{4,6} \\
0 & 0 & 0 & 0 & V_{5,5} & V_{5,6} \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

for some

\[
U_{2,3}, U_{2,4}, U_{2,5}, U_{3,4}, U_{3,5}, U_{4,5}, V_{2,3}, V_{2,4}, V_{2,5}, V_{3,4}, V_{3,5}, V_{4,5} \in B_n,
\]

\[
U_{1,2}, U_{1,3}, U_{1,4}, U_{1,5}, V_{1,2}, V_{1,3}, V_{1,4}, V_{1,5} \in B_{1 \times n},
\]

\[
U_{2,6}, U_{3,6}, U_{4,6}, U_{5,6}, V_{2,6}, V_{3,6}, V_{4,6}, V_{5,6} \in B_{n \times 1},
\]

\[
U_{1,6}, V_{1,6} \in B,
\]

\[
U_{2,2}, U_{3,3}, U_{4,4}, U_{5,5}, V_{2,2}, V_{3,3}, V_{4,4}, V_{5,5} \in U_{n}.
\]

The top right block of \( X \) and the block immediately below it are both equal to 0, so by monotonicity the same blocks in \( UY \) must be equal to 0. The top right
block of $UY$ is $U_{1,4}1^{n\times 1} + U_{1,5}1^{n\times 1} + U_{1,6}$, so $U_{1,4}, U_{1,5}$ and $U_{1,6}$ are all equal to 0. The block immediately below this is $U_{2,4}1^{n\times 1} + U_{2,5}1^{n\times 1} + U_{2,6}$, so $U_{2,4}, U_{2,5}$ and $U_{2,6}$ are all equal to 0.

We see that

\[
UY = \begin{bmatrix}
1 & 1 & 1 & U_{1,3}Y_0 & 0 & 0 \\
0 & U_{2,2}1^{n\times n} + U_{2,3} & U_{2,3}Y_0 & 0 & 0 \\
0 & 0 & U_{3,3} & U_{3,4}1^{n\times n} & U_{3,4}1^{n\times 1} + U_{3,5}1^{n\times 1} + U_{3,6} \\
0 & 0 & 0 & U_{4,4} & 1 & 1 \\
0 & 0 & 0 & 0 & U_{5,5} & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

The block immediately to the left of the top right block of $X$ is also equal to 0, and in the product $UYV$ this block is equal to $V_{1,5} + U_{1,3}Y_0 V_{4,5}$. Therefore $V_{1,5}, V_{2,5}$ and $V_{3,5}$ are all equal to 0. The block immediately below this is now equal to $X_0$ in $X$ and $U_{2,3}Y_0 V_{4,5}$ in $UYV$. Therefore there are some $U_0, V_0 \in \mathbb{B}_n$ such that $U_0Y_0V_0 = X_0$, as required.

$\square$
Chapter 6

Gossip Monoids

Some of the results in this chapter are joint work with Mark Kambites and Marianne Johnson and appear in the paper [12]. Lemma 6.13 and Theorems 6.6, 6.14, 6.15, and 6.16 are the result of this joint work. Although the theorems (and lemma) were originally proved by myself, the proofs which appear here and in the paper have been significantly improved by the contributions of Mark Kambites and Marianne Johnson.

The next class of monoids, the Gossip Monoids, are submonoids of $\mathbb{R}^n$ which relate to a concept in network communication known as gossiping. In [7] this concept is described as follows: “Every node of a network (processor) has a piece of information (value) which has to be transmitted to all other nodes by exchanging messages along the links of the network. ... In a single round (lasting one unit of time) every node can communicate with at most one neighbor, and during such a transmission communicating nodes exchange all the values they currently know.” This description makes obvious the applications in computer science and networking, and gossiping also has applications in epidemiology, but the concept originally arose in the form of a puzzle posed by A. Boyd and popularised by P. Erdős. In this puzzle, known as the gossip problem or the telephone problem,
gossiping people take the place of the processors and scandals take the place of the values.
Definition 6.1: Gossip Problem

Consider $n$ gossips, each of whom knows a unique scandal unknown to the others. The gossips communicate by phone and in a phone call the two participants tell each other every scandal they know. What is the minimum number of phone calls required before every gossip knows every scandal?

In this thesis we shall use the original context of gossip and speak in terms of gossiping people and scandals. The gossip problem was solved independently by Tijdeman [32], Baker and Shorstak [1], Hajnal, Milner and Szemeredi [16], and others, who showed in various different ways that the solution is:

\[
\begin{align*}
0 & \text{ if } n = 1, \\
1 & \text{ if } n = 2, \\
3 & \text{ if } n = 3, \\
2n - 4 & \text{ if } n \geq 4.
\end{align*}
\]

Gossip concerns states of knowledge among $n$ gossips and $n$ scandals. Such a state of knowledge can be represented by some $A \in \mathbb{B}_n$ as in [4] (which uses $\left(\{\infty, 0\}, \min, +\right)$ as its Boolean semiring) as follows:

\[
a_{i,j} = \begin{cases} 
1 & \text{if gossip } j \text{ knows scandal } i, \\
0 & \text{otherwise.}
\end{cases}
\]

Example 6.2

Consider a situation with five gossips and five scandals. Gossip 1 knows the first and third scandals, gossip 2 knows the second and fourth scandals, gossip 3 knows the first scandal, gossip 4 knows the fifth scandal and gossip 5 knows the second and fifth scandals. This state of knowledge is represented by the matrix
We can represent a phone call between two gossips with right multiplication by a particular type of Boolean matrix:

**Definition 6.3: Phone Call Matrix**

Given \(a, b \leq n\), let \(C[a, b] \in \mathbb{B}_n\) be the matrix defined by

\[
c[a, b]_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } \{i, j\} = \{a, b\}, \\
0 & \text{otherwise.}
\end{cases}
\]

We call matrices of this form phone call matrices.

**Example 6.2 (continued)**

In \(\mathbb{B}_5\) the phone call matrix \(C[3, 5]\) is as follows:

\[
C[3, 5] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}.
\]

All phone call matrices have this general form - they have 1s on the diagonal and only two off-diagonal 1s, located opposite each other.

If \(B = AC[a, b]\) for some \(A, B \in \mathbb{B}_n\), then every column of \(B\) is the same as the corresponding column of \(A\) except for columns \(a\) and \(b\). Columns \(a\) and \(b\) of \(B\) are equal to each other and are both equal to the Boolean sum of columns \(a\) and \(b\) in \(A\).
In terms of gossips and scandals, in the state of knowledge $B$ each gossip knows the same set of scandals as they did in state of knowledge $A$, except for gossips $a$ and $b$. In $B$, gossips $a$ and $b$ both know every scandal which was known to either $a$ or $b$ in $A$. In other words, $A$ and $B$ represent states of knowledge before and after a phone call between $a$ and $b$.

Similarly, if $B = C[a, b]A$ then rows $a$ and $b$ of $B$ are both equal to the sum of rows $a$ and $b$ from $A$, and each other row is the same in $B$ as it is in $A$.

**Example 6.2 (continued)**

Let $A \in \mathbb{B}_5$ be the same as before:

$$A = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix},$$

Then the product $AC[3, 5]$ is the result of replacing columns 3 and 5 of $A$ with their sum while leaving the other columns unchanged:

$$AC[3, 5] = \begin{bmatrix}
1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1
\end{bmatrix}.$$

This matrix represents the state of knowledge obtained by a phone call between gossips 3 and 5 after starting at the state represented by $A$. The product $C[3, 5]A$ is the result of replacing row 3 and 5 of $A$ with their sum while leaving the other rows unchanged:
\[
C[3, 5]A = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

For fixed \( n \), gossiping involves starting with the state of knowledge represented by the \( n \times n \) identity matrix, and then applying phone calls, which are represented by right multiplication by \( n \times n \) phone call matrices. The monoid generated by the \( n \times n \) phone call matrices therefore contains all states of knowledge that can be obtained through gossiping.

**Definition 6.4: Gossip Monoid**

The gossip monoid of rank \( n \), denoted \( G_n \), is the monoid generated by all phone call matrices of order \( n \):

\[
G_n = \langle \{C[a, b] : a, b \leq n\} \rangle.
\]

Each generator of each gossip monoid is clearly reflexive, so each gossip monoid is a submonoid of the reflexive monoid of the same size.

In [4] Brouwer, Draisma and Frenk define a generalisation of gossip monoids called lossy gossip monoids. They work over the semiring \([0, \infty], \min, +\) with additive identity \( \infty \) and multiplicative identity 0. The generators of the lossy gossip monoid of rank \( n \) are the \( n \times n \) matrices \( C_{k,l}(a) \) for \( k, l \leq n \) and \( a \in [0, \infty] \) defined by

\[
c_{k,l}(a)_{i,j} = \begin{cases} 
0 & \text{if } i = j, \\
a & \text{if } \{i, j\} = \{k, l\}, \\
\infty & \text{otherwise}.
\end{cases}
\]

They also define the ordinary gossip monoid of rank \( n \) to be the submonoid of the lossy gossip monoid of rank \( n \) generated by the elements \( C_{k,l}(0) \) for \( k, l \leq n \). This monoid is functionally identical to our gossip monoid of rank \( n \). In [4] they...
prove that the maximum number of phone call matrices in an irredundant product is \( n(n - 1)/2 \), and this bound is attained. An earlier version of the paper contained an incorrect proof of this theorem, which is replaced in the current version with an alternative proof. We shall prove the theorem here using a fixed version of the original proof.

**Theorem 6.5**

Any sequence of phone calls among \( n \) gossiping parties such that in each phone call both participants exchange all they know, and at least one of the parties learns something new, has length at most \( \left( \binom{n}{2} \right) \), and this bound is attained.

**Proof.** The following sequence of \( \left( \binom{n}{2} \right) \) phone calls shows that the claimed maximum is attained. First gossip 1 calls each gossip from gossip 2 to gossip \( n \), in turn. Next gossip 2 calls each gossip from gossip 3 to gossip \( n \), in turn. This continues until gossip \( n - 1 \) calls gossip \( n \). In each call the first participant learns the second participant’s scandal, so at least one party learns something new.

To show that \( \binom{n}{2} \) is an upper bound we consider a maximal sequence of irredundant phone calls. Number the phone calls in increasing order. In each call at least one piece of information is transferred. When a call transfers \( 1 + e \) pieces of information we say that \( e \) extra pieces of information were transferred on that call. The total number of pieces of information learned, across all calls, is therefore equal to the number of calls plus the total number of extra items.

We shall assign a call from the sequence to each ordered pair \((a, b)\) of distinct integers between 1 and \( n \) in such a way that a call with \( e \) extra items will have no more than \( 2e \) pairs assigned to it. In other words, a call with \( p \) pairs assigned to it will have at least \( p/2 \) extra items. Then, since the number of ordered pairs of distinct integers between 1 and \( n \) is \( n(n - 1) \), the number of extra items across all calls is at least \( n(n - 1)/2 \). Since the sequence of phone calls is maximal each of the \( n \) gossips
learns each of the other $n - 1$ scandals, so the total number of items learned is $n(n - 1)$. The number of calls is therefore no more than $n(n - 1) - n(n - 1)/2 = \binom{n}{2}$.

Let $a \neq b$. We will define the canonical path of scandal $a$ to gossip $b$ as the maximal strictly increasing sequence of calls $(i_1, i_2, \ldots, i_h)$ which satisfies the following: there is some sequence of indices $(x_0, x_1, \ldots, x_h)$ with $x_h = b$ such that for each $1 \leq j \leq h$, call $i_j$ is between gossip $x_{j-1}$ and gossip $x_j$, and gossip $x_j$ knows both scandals $a$ and $b$ after call $i_j$ but does not know both of them before the call. We use the notation $(a \rightarrow b)$ for the canonical path of scandal $a$ to gossip $b$. The last call in $(a \rightarrow b)$ must be the call where gossip $b$ learns scandal $a$, and the first call in $(a \rightarrow b)$ must be a call where two gossips exchange scandals $a$ and $b$.

We order the canonical paths using reverse lexicographic (revlex) ordering. This is lexicographic ordering on the reversed paths. When one path is the tail of another, we take the longest path to be the revlex smallest.

For each call $i$ and each gossip $b$, let $B_i$ be the subset of $\{1, 2, \ldots, n\}$ indexing the scandals learned by gossip $b$ on call $i$. For each non-empty $B_i$ let $a_0$ be an element of $B_i$ which revlex minimises $(a_0 \rightarrow b)$. Let $(a_0 \rightarrow b) = (i_1, i_2, \ldots, i_h)$ and note that we must have $i_h = i$. We assign the pair $(a_0, b)$ to the call $i_1$ and call this the initial assignment. We assign $(a, b)$ to call $i_h = i$ for all other $a \in B_i$ and call this the excess assignment. Since each gossip learns each other gossip’s scandal, for each $b$ each scandal other than $b$ will be in $B_i$ for some $i$, so each pair of distinct integers will be assigned to a call.

We must now check that a call $j$ with $e$ extra items has no more than $2e$ pairs assigned to it. Let gossips $p$ and $q$ be the participants in call $j$, let $s = |P_j|$ and let $t = |Q_j|$. Then the number of items learned in call $j$ is $s + t$, so $e = s + t - 1$ we are allowed to assign $2(s + t - 1)$ pairs to call $j$.

We first consider how many pairs are assigned to $j$ as the result of an excess assignment. If $(a, b)$ is assigned to $j$ in an excess assignment, then $j$ is the last call
in \((a \rightarrow b)\), so either \(b = p\) and \(a \in P_j\) or \(b = q\) and \(a \in Q_j\). One pair \((a_0, p)\) with \(a_0 \in P_j\) will be assigned somewhere in an initial assignment and the other \(s - 1\) pairs \((a, p)\) with \(a \in P_j\) will be assigned to \(j\) in an excess assignment. Similarly, \(t - 1\) pairs of the form \((a, q)\) with \(a \in Q_j\) are assigned to \(j\) in an excess assignment.

We next consider how many pairs are assigned to \(j\) from initial assignments. If \((a, b)\) is assigned to \(j\) in an initial assignment then \(j\) is the first call in \((a \rightarrow b)\), so \(j\) is a phone call where two gossips exchange scandals \(a\) and \(b\). Thus either \(a \in P_j\) and \(b \in Q_j\) or \(a \in Q_j\) and \(b \in P_j\).

For a fixed \(f \in Q_j\) and any two \(c, d \in P_j\) we claim that if \((c \rightarrow f)\) and \((d \rightarrow f)\) both contain \(j\) then they are identical. For contradiction, assume \((c \rightarrow f)\) and \((d \rightarrow f)\) do both contain \(j\) but are not identical. Since \(j\) is a call where scandals \(c\) and \(f\) are exchanged, and it is in \((c \rightarrow f)\), it must be the first call in \((c \rightarrow f)\). Let \((c \rightarrow f) = (i_1, i_2, \ldots, i_h)\) and note that \(i_1 = j\). Since scandal \(d\) is learned by gossip \(p\) in call \(j\), we know that for each call from \(i_1\) to \(i_h\) both participants know scandal \(d\) at the end of the call. Since this is the canonical path of scandal \(c\) to gossip \(f\) we also know that both participants know scandal \(f\) at the end of each call. Thus at the end of each call both participants know both scandals \(d\) and \(f\), but this is not the canonical path of scandal \(d\) to gossip \(f\), so there must be some call on this path where both participants know both scandals \(d\) and \(f\) before the call. This means that \((d \rightarrow f)\) is strictly revlex smaller than \((c \rightarrow f)\). Applying the same reasoning with the roles of \(c\) and \(d\) reversed we find that each canonical path is strictly smaller than the other, giving the required contradiction.

This claim tells us that for a fixed \(f \in Q_j\) there is at most one unique canonical path \((a \rightarrow f)\) such that \(a \in P_j\), and so no more than one pair of the form \((a, f)\) is assigned to call \(j\) in an initial assignment. Therefore no more than \(t\) pairs \((a, b)\) with \(a \in P_j\) and \(b \in Q_j\) can be assigned to call \(j\) in an initial assignment. Similarly, no more than \(s\) pairs \((a, b)\) with \(a \in Q_j\) and \(b \in P_j\) can be assigned to call \(j\) in an
initial assignment.

This covers all possible assignments of pairs to call \( j \), so the total number of pairs assigned to call \( j \) is at most \((s - 1) + (t - 1) + t + s = 2(s + t - 1)\), as required to complete the proof.

Each phone call matrix is an idempotent, so the gossip monoid is an idempotent generated monoid. The generators are not the only idempotents though. The following theorem characterises all of the idempotents.

**Theorem 6.6**

The idempotents of \( G_n \), the gossip monoid of rank \( n \), are precisely the matrices \( A \in \mathbb{B}_n \) which satisfy \( a_{i,j} = 1 \iff i \sim j \) for some equivalence relation \( \sim \).

**Proof.** Let \( A \in G_n \) be a gossip matrix. Then \( A = \prod_{r=0}^{m} C[a_r, b_r] \) for some phone call matrices \( C[a_0, b_0], \ldots, C[a_m, b_m] \).

Let \( A \) be idempotent and define \( \sim_A \) by \( i \sim_A j \iff a_{i,j} = 1 \). Since \( a_{i,i} = 1 \) for all \( i \), \( \sim_A \) is reflexive. Since \( A \) is idempotent and each phone call matrix is symmetric,

\[
A = A^{m+1} = \prod_{t=m}^{0} A
= \prod_{t=m}^{0} \prod_{r=0}^{m} C[a_r, b_r]
\geq \prod_{t=m}^{0} C[a_t, b_t]
= \prod_{t=m}^{0} C[a_t, b_t]^T
= \left( \prod_{t=0}^{m} C[a_t, b_t] \right)^T
= A^T.
\]
Thus $A = A^T$ and so $\sim_A$ is symmetric. Since $A = A^2$ we have, for each $i, j \leq n$,

$$a_{i,j} = \sum_{k=1}^{n} a_{i,k} a_{k,j},$$

so

$$i \sim_A k \text{ and } k \sim_A j \Rightarrow a_{i,k} = a_{k,j} = 1$$
$$\Rightarrow a_{i,j} = 1$$
$$\Rightarrow i \sim_A j,$$

and $\sim_A$ is transitive.

This shows that if $A$ is an idempotent of the gossip monoid of rank $n$ then it satisfies $a_{i,j} = 1 \iff i \sim j$ for some equivalence relation $\sim$.

For the converse, let $\sim$ be an equivalence relation and define $A$ by $a_{i,j} = 1 \iff i \sim j$. We first show that $A \in G_n$. If $\sim$ is equality then $A$ is the $n \times n$ identity matrix, which is an element of $G_n$. Otherwise there is at least one pair $i \neq j$ such that $i \sim j$ and we will show that $A$ can be written as the product of all the phone call matrices $C[i, j]$ such that $i \sim j$, taken in any order.

Let $P$ be a product of all phone call matrices $C[i, j]$ such that $i \sim j$, in some order. For each $i, j$ such that $a_{i,j} = 1$ we have $C[i, j]_{i,j} = 1$, so $p_{i,j} = 1$ by monotonicity. Thus $A \preceq P$.

Next we show that if $B \preceq A$ and $C \preceq A$ then $BC \preceq A$. If $(BC)_{i,j} = 1$ then there is some $k$ such that $b_{i,k} = c_{k,j} = 1$, so $a_{i,k} = a_{k,j} = 1$ and so $i \sim k$ and $k \sim j$. By transitivity $i \sim j$ so $a_{i,j} = 1$. This shows that $BC \preceq A$. By symmetry, $C[i, j] \preceq A$ for each $i, j$ such that $i \sim j$, so by induction $P \preceq A$, and so $A = P$. Therefore $A \in G_n$.

By monotonicity we have $A \preceq A^2$ and by the same reasoning as in the previous paragraph, since $A \preceq A$ we have $A^2 \preceq A$. Thus $A = A^2$ and so $A$ is idempotent. \qed
Returning to the context of gossips and scandals, we note that if a gossip doesn’t know anybody else’s scandals it is because they haven’t talked to anybody, and if they haven’t talked to anybody then nobody knows their scandal. The following lemma formalises this:

**Lemma 6.7**

Let \( A \in G_n \) and fix some \( j \leq n \). If column \( j \) of \( A \) is equal to the \( j \)th unit vector then row \( j \) of \( A \) is also equal to the \( j \)th unit vector.

**Proof.** Let \( A \in G_n \) be such that column \( j \) of \( A \) is equal to the \( j \)th unit vector. For any phone call matrix \( C[i, j] \) with \( i \neq j \), we have \( c[i, j]_{i,j} = 1 \) so by monotonicity \( C[i, j] \) cannot appear in any factorisation of \( A \). Left multiplication by a matrix \( C[k, l] \) changes rows \( k \) and \( l \) but leaves all other rows unchanged. Therefore, since \( C[i, j] \) is not a factor of \( A \) for any \( i \), left multiplication of \( I_n \) by \( A \) leaves row \( j \) unchanged, and so row \( j \) of \( A \) is equal to the \( j \)th unit vector. \( \square \)

Whenever a gossip learns something they share everything they know with the person teaching them. Therefore the last person to talk to them is guaranteed to know everything they do. As a result, for each gossip there is some other gossip who knows everything that they do, and the only exception is if a gossip has not participated in any phone calls. The following lemma formalises this:

**Lemma 6.8**

Let \( A \in G_n \) and fix some \( j \leq n \). If, for each \( i \neq j \), column \( i \) of \( A \) is not \( \preceq \)-above column \( j \) of \( A \), then column \( j \) is the \( j \)th unit vector.
Proof. Write $A \in G_n$ as a product of phone call matrices $A = C_1C_2 \cdots C_m$ and assume column $j$ of $A$ is not equal to the $j$th unit vector. We will show that there is some other column of $A$ which is $\preceq$-above column $j$.

Since right multiplication by $C[a,b]$ changes columns $a$ and $b$ but leaves the other columns unchanged, and right multiplying $I_n$ by $C_1C_2 \cdots C_m$ changes column $j$, some phone call matrix in this product must be equal to $C[i,j]$ for some $i$. Let $C_p = C[k,j]$ be the last phone call matrix involving $j$, so that column $j$ of $A$ is equal to column $j$ of $C_1C_2 \cdots C_p$. Since $C_p = C[k,j]$, columns $k$ and $j$ are equal to each other in $C_1C_2 \cdots C_p$, and by monotonicity column $k$ of $A$ is $\preceq$-above column $k$ of $C_1C_2 \cdots C_p$. Therefore column $k$ of $A$ is $\preceq$-above column $j$ of $A$, as required.

We can also represent a conference call between a set of gossips with a particular type of gossip matrix.

**Definition 6.9: Conference Call Matrix**

Given $S \subseteq \{1, \ldots, n\}$, let $C[S] \in B_n$ be the matrix defined by

$$c[S]_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ or } i, j \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We call matrices of this form conference call matrices.

It is clear that the binary relation associated with $C[S]$ is an equivalence relation, so $C[S] \in G_n$ as claimed and in particular it is an idempotent. It is not hard to see that if $S$ and $T$ are disjoint sets then $C[S]$ and $C[T]$ commute. In particular if $\{i,j\}$ and $\{k,l\}$ are disjoint sets then $C[i,j]$ and $C[k,l]$ commute.

The next theorem shows that any state of knowledge which can be obtained through gossip can be obtained in such a way that in each call every participant learns something new, so long as conference calls are allowed.
Theorem 6.10
Any \( A \in G_n \) can be written as a product of conference call matrices \( A = \prod_{i=1}^{r} C[S_i] \) such that for each \( 1 \leq k \leq r \), each column of \( \prod_{i=1}^{k} C[S_i] \) indexed by \( S_k \) is strictly \( \preceq \)-above the corresponding column of \( \prod_{i=1}^{k-1} C[S_i] \).

Proof. Let \( A \in G_n \). By definition, \( A \) can be written as a product of phone call matrices. Let \( A = \prod_{i=1}^{m} C[a_i, b_i] \) be such a product.

We will modify the sequence \( C[a_i, b_i] \) to create the desired product of conference calls by repeating the following process until no call remains in which some participant learns nothing. The process contains several claims which we will prove.

1) Look for the first call where a participant learns nothing. We claim that this call will contain just two participants. If neither participant learns anything, simply remove the call and restart the process, since this clearly does not change the product. Otherwise just one participant learns nothing. Let this be call \( j \) and call the participants \( a \) and \( b \), where \( a \) is the participant who learns nothing.

2) Since \( a \) learns nothing in call \( j \), they must already know scandal \( b \) before the call. Therefore \( a \) has been involved in at least one call before call \( j \). Let \( i \) be the last call involving gossip \( a \) before call \( j \), and let \( S \) be the set of participants in this call other than \( a \).

3) For each call strictly between \( i \) and \( j \), in order, check if it shares any participants with call \( i \) or with any of the calls between \( i \) and itself. If not, reorder the product by moving this call to directly before call \( i \). If two calls have no participants in common then they commute, so this reordering does not change the result of the product.
4) We claim that none of the remaining calls strictly between \( i \) and \( j \) share any participants with call \( j \). Reorder the product by moving call \( j \) to directly after call \( i \). Again, by commutativity the result of the product is unaffected.

5) Replace calls \( i \) and \( j \) with a conference call between \( a, b \), and everyone in \( S \).

We claim that this does not affect the product and that every participant in this call learns something new.

We now verify that the claims made in (1), (4) and (5) are correct.

In step (5) we replace \( C[\{a\} \cup S]C[a,b] \) with \( C[\{a,b\} \cup S] \). Let \( T_x \) be the set of items known to gossip \( x \) before these calls. After call \( C[\{a\} \cup S] \), gossip \( a \) and each gossip in \( S \) knows \( T_a \cup \bigcup_{x \in S} T_x \), then after call \( C[a,b] \) gossips \( a \) and \( b \) know \( T_a \cup T_b \cup \bigcup_{x \in S} T_x \). Since \( a \) learned nothing new in the second call we have \( T_a \cup \bigcup_{x \in S} T_x = T_a \cup T_b \cup \bigcup_{x \in S} T_x \), so all gossips in \( \{a,b\} \cup S \) know \( T_a \cup T_b \cup \bigcup_{x \in S} T_x \) at the end of these calls. This is precisely the outcome of call \( C[\{a,b\} \cup S] \) so the replacement, as claimed, does not change the result of the product.

Every gossip in \( \{a\} \cup S \) learns something new in call \( i \), otherwise call \( j \) would not have been the first call where somebody learns nothing. Also, \( b \) learns something new in call \( j \), otherwise neither participant would have learned anything. Since call \( C[\{a,b\} \cup S] \) has the same result as call \( i \) followed by call \( j \), each participant learns something new in this new call. This proves the claim in step (5).

The claim in step (1) now follows since the only calls with more than two participants are those created in step (5), and each participant in these calls learns something new. Although steps (3) and (4) reorder some calls, all reordering is done by swapping pairs of adjacent calls with disjoint sets of participants. Swapping the order of two such calls does not change the information learned in either call, and thus it remains true that in each call with more than two participants, each participant learns something new.
Finally, for contradiction assume the claim in step (4) is false, so there is some call, \( k \), between calls \( i \) and \( j \) which shares a participant with \( j \). Since call \( k \) is between \( i \) and \( j \) it does not involve \( a \) (by the definition of \( i \) in step (2)). Thus it involves \( b \).

Since call \( k \) is still between calls \( i \) and \( j \) after step (3), it shares a participant with either call \( i \) or some call between calls \( i \) and \( k \). If it shares a participant with some call between calls \( i \) and \( k \) then this call, in turn, shares a participant with either \( i \) or some call between \( i \) and itself, and so on. Since there is only a finite number of calls between \( i \) and \( k \), there is a finite subsequence of calls, starting at \( i \) and ending at \( k \), such that each call in the subsequence shares a participant with the next.

The first call in the subsequence is call \( i \), so it involves gossip \( a \), so after the call each participant knows \( T_a \cup \bigcup_{x \in S} T_x \). One of these participants is in the second call, so at the end of the second call each participant knows all information in \( T_a \cup \bigcup_{x \in S} T_x \). This continues until the last call, call \( k \), which involves gossip \( b \). Thus gossip \( b \) knows \( T_a \cup \bigcup_{x \in S} T_x \) before call \( j \) and therefore learns nothing new in call \( j \), a contradiction.

This proves all the claims. Each iteration of the process reduces the number of calls in which nothing is learned. Therefore there will eventually be no such calls remaining and we will have a product of conference calls such that in each call every participant learns something new.

Since the gossip monoids are defined by generators rather than by the structure of the matrix, in general it is not obvious whether or not a given \( n \times n \) matrix is an element of \( G_n \). The gossip monoid’s membership problem is therefore of interest:

**Definition 6.11: Gossip Membership Problem (GMP)**

*Given \( C \in \mathbb{B}_n \), the gossip membership problem asks if \( C \in G_n \).*
We also consider the following decision problem, which will function as an intermediate step in a polynomial time reduction:

**Definition 6.12: Maximal Gossip Transformation Problem (MGTP)**

*Given* $A, B \in \mathbb{B}_n$ *such that* $A$ *satisfies the maximal column condition: $A$ is non-zero and the set of distinct columns of $A$ form an anti-chain* (In other words, every column is non-zero and maximal among the set of columns), *the maximal gossip transformation problem asks if there exists* $T \in G_n$ *such that* $AT = B$.

We will require the following lemma, which allows us to “nest” an arbitrary Boolean matrix inside a gossip matrix with polynomially larger size:

**Lemma 6.13**

*Let* $n \geq 2$. *For any* $A \in \mathbb{B}_n$, *the following matrix lies in* $G_{n^2+n}$:

$$X = \begin{pmatrix} \begin{array}{c|c} \begin{array}{c} n^2 \\ \hline \end{array} & \begin{array}{c} n \\ \hline \end{array} \\ \hline \begin{array}{c} \mathbf{1} \\ \hline \end{array} & A \\ \hline \begin{array}{c} \mathbf{1} \\ \hline \end{array} & \mathbf{1} \\ \hline \end{array} \end{pmatrix}^{n^2}.$$

**Proof.** We shall show that $X$ can be written as $X = X_1X_2X_3X_4$, where the matrices $X_i$ are defined in terms of conference call matrices as follows:

$$X_1 = C[\{n + 1, \ldots, n^2 + n\}], \quad X_2 = \prod_{i=1}^{n} C[\{n(j - 1) + i : 1 \leq j \leq n\}],$$

$$X_3 = \prod_{j=1}^{n} C[\{n(j - 1) + i : a_{i,j} = 1\} \cup \{n^2 + j\}], \quad X_4 = C[\{1, \ldots, n^2\}],$$

and where the products taken in $X_2$ and $X_3$ can be in any order since the defining sets are disjoint.

Each matrix in the product $X_2$ is of the form $C[\{n(j - 1) + i : 1 \leq j \leq n\}]$ for some $i$. This matrix has a 1 in the $a, b$ position if and only if either $a = b$, or
$a, b \leq n^2$ and $a$ and $b$ are both congruent to $i$ modulo $n$. The product $X_2$ therefore has a 1 in the $a, b$ position if and only if either $a = b$, or $a, b \leq n^2$ and $a$ and $b$ are congruent to each other modulo $n$. The result is that the top-left $n^2 \times n^2$ block of $X_2$ consists of an $n \times n$ array of copies of $I_n$:

$$X_2 = \begin{bmatrix} n^2 & n \\ I_n & \ldots & I_n \\ \vdots & \ddots & \vdots \\ I_n & \ldots & I_n \\ n & \ldots & 0 \\ 0 & \ldots & I_n \end{bmatrix}_{n^2 \times n}.$$

For $1 \leq j \leq n$ let $S_j = \{ i : a_{i,j} = 1 \}$ and let $A_{[j]}$ be the $n \times n$ matrix which is the same as $A$ on column $j$ but zero elsewhere. For each $j$ we have

$$C[\{n(j-1) + i : a_{i,j} = 1\} \cup \{n^2 + j\}] = \begin{bmatrix} n(j-1) & n & n(n-j) & n \\ I_{n(j-1)} & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & I_{n(n-j)} & 0 \\ 0 & (A_{[j]})^T & 0 & I_n \end{bmatrix}_{n \times n}.$$

The product of all such matrices, as $j$ ranges between 1 and $n$, is
Consider the effect of multiplying $X_3$ on the left by $X_2$. This leaves the last $n$ rows unchanged. The remaining $n^2$ rows can be split into $n$ blocks of $n$ consecutive rows each. The structure of the identity matrices in $X_2$ means that in $X_2X_3$ these blocks are all identical to each other and equal to the sum of the $n$ corresponding blocks in $X_3$. Since the sum of $A_1$ to $A_n$ is $A$, we get

$$X_2X_3 = \begin{bmatrix} \text{Block 1} & \cdots & \text{Block } n \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ \vdots & \ddots & \vdots & \vdots \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ (A_1)^T \vdots (A_2)^T \vdots \vdots (A_n)^T & I_n \\ \end{bmatrix} \begin{bmatrix} \text{Block 1} & \cdots & \text{Block } n \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ \vdots & \ddots & \vdots & \vdots \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ (A_1)^T \vdots (A_2)^T \vdots \vdots (A_n)^T & I_n \\ \end{bmatrix} \begin{bmatrix} \text{Block 1} & \cdots & \text{Block } n \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ \vdots & \ddots & \vdots & \vdots \\ C[S_1] \vdots C[S_2] \vdots \vdots C[S_n] & A_1 \vdots A_2 \vdots \vdots A_n \\ (A_1)^T \vdots (A_2)^T \vdots \vdots (A_n)^T & I_n \\ \end{bmatrix}$$

Between them, the last $n^2$ rows of $X_2X_3$ contain a 1 in each column (each conference call has only 1s on its diagonal). Therefore, multiplying on the left by $X_1 = C[\{n + 1, \ldots, n^2 + n\}]$ fills each of these rows with 1s. Now, between them, the first $n^2$ columns of $X_1X_2X_3$ contain a 1 in each row, so multiplying on the right by $X_4 = C[\{1, \ldots, n^2\}]$ fills each of these columns with 1s. We therefore obtain $X_1X_2X_3X_4 = X$, as required. \qed
Theorem 6.14

The decision problem MGTP is NP-hard.

Proof. Given $A, B \in \mathbb{B}_n$ such that $A$ satisfies the maximal column condition, MGTP asks if there exists $T \in G_n$ such that $AT = B$. We will show MGTP is NP-hard via a polynomial time reduction from DSP.

Let $H = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$, and let $k \in \mathbb{N}$ be such that $0 < k < n$. We will construct matrices $A, B \in \mathbb{B}_{3n}$, with $A$ satisfying the maximal column condition, such that $AT = B$ for some $T \in G_{3n}$ if and only if $H$ admits a dominating set of size $k$.

Recall from the proof of Theorem 2.7 that if we regard $H$ as a Boolean matrix then there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $(H + I_n)v = 1$ if and only if the graph $H$ has a dominating set of size $k$. MGTP concerns matrices which satisfy the maximal column condition, so we will construct a matrix which satisfies the maximal column condition but is similar enough to $H + I_n$ to be used in its place.

For each $i \leq n$, there is some maximal column of $H + I_n$ which is $\preceq$-above or equal to column $i$ of $H + I_n$. Define a function $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that for each $i \leq n$, column $f(i)$ is $\preceq$-above or equal to column $i$. We can then define a matrix $M$ which satisfies the maximal column condition by letting column $i$ of $M$ be equal to column $f(i)$ of $H + I_n$.

If there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $(H + I_n)v = 1$ then since $M \succeq H + I_n$ we have $Mv \succeq (H + I_n)v = 1$, so there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $Mv = 1$. Conversely if there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $Mv = 1$ then any vector $u \in \mathbb{B}^n$ which satisfies $u_{f(i)} \geq v_i$ for each $i \leq n$ also satisfies $(H + I_n)u \succeq Mv = 1$. The condition $u_{f(i)} \geq v_i$ does not require $u$ to have more than $k$ 1s, so it is possible to choose such a $u$ with exactly $k$ 1s. Therefore there exists a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $(H + I_n)v = 1$ if and only if there exists a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $Mv = 1$. 
Note that finding the non-maximal columns of $H + I_n$ and finding maximal columns $\preceq$-above them only requires $n(n - 1)/2$ vector comparisons, so $M$ can be computed in polynomial time.

Let $A$ and $B$ be the $3n \times 3n$ Boolean matrices

$$A = \begin{bmatrix} n & n & n \\ M & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}^n, \quad B = \begin{bmatrix} n & n & k \ n - k \\ M & M & 1 \ 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}^n.$$

Note that the matrix $A$ satisfies the maximal column condition since $M$ does, and $A$ and $B$ give an instance of the problem MGTP.

If $H$ admits a dominating set of size $k$ then there is a vector $v \in \mathbb{B}^n$ with $k$ 1s such that $Mv = 1$. Let $j_1, j_2, \ldots, j_k$ index the $k$ 1s in $v$, so that (where $e_i$ is the $i$th unit vector)

$$\sum_{i=1}^k e_{j_i} = v.$$

We will define a $T \in G_{3n}$ such that $AT = B$ by writing it as a product of phone call matrices. Let $T_1$, $T_2$ and $T_3$ be

$$T_1 = \prod_{i=1}^n C[i, n + i], \quad T_2 = \prod_{i=1}^k C[j_i, 2n + i],$$

$$T_3 = C[\{2n + 1, 2n + 2, \ldots, 2n + k\}],$$

where the products can be taken in any order since the terms commute with each other. We then define $T = T_1 T_2 T_3 \in G_{3n}$. Now we have

$$T_1 = \begin{bmatrix} n & n & n \\ I_n & I_n & 0 \\ I_n & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix}^n,$$
\[
T_1T_2 = \begin{bmatrix}
I_n & I_n & e_{j_{k}} & 0 \\
I_n & I_n & e_{j_{k}} & 0 \\
e_{j_1} & 0 & I_n \\
e_{j_2} & 0 & 0 & I_{n-k}
\end{bmatrix}
\]

and since \( \sum_{i=1}^{k} e_{j_i} = \mathbf{v} \),
\[
T = T_1T_2T_3 = \begin{bmatrix}
I_n & I_n & \mathbf{v} & 0 \\
I_n & I_n & \mathbf{v} & 0 \\
e_{j_1} & 0 & 1 & 0 \\
e_{j_2} & 0 & 0 & I_{n-k}
\end{bmatrix}
\]

Then since \( M\mathbf{v} = 1 \) it is easy to verify that \( AT = B \).

Conversely, if there is a \( T \in G_{3n} \) such that \( AT = B \), then there is a sequence of phone call matrices whose product is \( T \), say \( T = C_1C_2 \cdots C_q \). We may clearly assume without loss of generality that the sequence contains no redundant factors, in the sense that \( C_1 \cdots C_p \neq C_1 \cdots C_p C_{p+1} \) (which by monotonicity means that \( C_1 \cdots C_p \prec C_1 \cdots C_p C_{p+1} \)) for all \( p \).
Since right multiplication by a phone call matrix has the effect of replacing two columns with their sum, it is clear that each column of each product $AC_1 \cdots C_p$ must be equal to the sum of some collection of columns of $A$. In particular, the first $n$ rows of each column must equal the sum of some collection of columns of $M$. In the product $B = AC_1 \cdots C_q$ the first $n$ rows of column $n+i$, for $1 \leq i \leq n$, equal the $i$th column of $M$. By monotonicity and the fact that $M$ satisfies the maximal column condition, the first $n$ rows of column $n+i$ must equal either $0$ or the $i$th column of $M$ for every product $AC_1 \cdots C_p$.

Claim. For each $r \in \{2n+1, \ldots, 2n+k\}$, there is at most one matrix $C_p$ of the form $C[i,r]$ such that $i \leq 2n$ and the two products $AC_1 \cdots C_{p-1}$ and $AC_1 \cdots C_p$ differ on column $r$.

Proof of claim: Assume for contradiction that the sequence $C_1, \ldots, C_q$ contains phone call matrices $C_s = C[i, r]$ and $C_t = C[j, r]$, in that order, with $i, j \leq 2n$ and $2n+1 \leq r \leq 2n+k$, such that the products $AC_1 \cdots C_{s-1}$ and $AC_1 \cdots C_s$ differ on column $r$ and the products $AC_1 \cdots C_{t-1}$ and $AC_1 \cdots C_t$ also differ on column $r$.

We observe that the last $2n$ rows of column $r$ are identical in $A$ and $B$, so by monotonicity the differences in column $r$ occur in the first $n$ rows. Let $v_s, v_t \in \mathbb{B}^n$ be the first $n$ rows of column $r$ in the matrices $AC_1 \cdots C_s$ and $AC_1 \cdots C_t$, respectively. Since the differences occur in the first $n$ rows, and by monotonicity, we have $0 \prec v_s \prec v_t$. As the last phone call matrix in the product $AC_1 \cdots C_s$ is $C_s = C[i, r]$, columns $i$ and $r$ must be identical in this matrix, and so $v_s$ is also equal to the first $n$ rows of column $i$. If $i \leq n$ then $v_s$ is equal to column $i$ of $M$. Otherwise $n < i \leq 2n$ and since $v_s$ is non-zero it is equal to the $(i-n)$th column of $M$. Either way, $v_s$ is equal to a column of $M$ and similar observations show that $v_t$ is also equal to a column of $M$. Now $v_s \prec v_t$ contradicts the maximal column condition on $M$, and this proves the claim. □
It follows from the claim that right-multiplying \( A \) by the product \( C_1 \cdots C_q \) copies at most \( k \) of the columns of \( M \) into the columns \( 2n+1, \ldots, 2n+k \). Since the result of this right multiplication is to place a 1 in each of the first \( n \) rows of these columns, it must be that 1 is a sum of \( k \) columns of \( M \). There is therefore a vector \( v \in \mathbb{B}^n \) with \( k \) 1s such that \( Mv = 1 \). It follows from our observations that \( H \) admits a dominating set of size \( k \).

\[ \square \]

**Theorem 6.15**

The decision problem GMP is NP-complete.

**Proof.** Given \( C \in \mathbb{B}_n \), GMP asks if \( C \in G_n \).

By Theorem 6.5 each element of \( G_n \) can be written as a product of no more than \( n(n-1)/2 \) phone call matrices. Thus a non-deterministic Turing machine can guess an element of \( G_n \) in polynomial time and check if it is equal to \( C \). This shows that GMP is in NP. We show it is NP-hard via a polynomial time reduction from MGTP.

Let \( A, B \in \mathbb{B}_n \) with \( A \) satisfying the maximal column condition. We will define a Boolean matrix \( C \in \mathbb{B}_{n(n+4)} \), which can be constructed in polynomial time, such that \( C \in G_{n(n+4)} \) if and only if \( AT = B \) for some \( T \in G_n \). If \( n = 1 \) then we simply set \( C = I_5 \) if \( A = B \) and let \( C \) be the \( 5 \times 5 \) zero matrix if \( A \neq B \). Suppose, then, that \( n \geq 2 \) and let \( C \) be the \( n(n+4) \times n(n+4) \) matrix

\[
C = \begin{bmatrix}
1 & A & A & 0 & B \\
1 & 1 & 1 & 0 & 1 \\
0 & 0 & I_n & 1 & 1 \\
0 & I_n & 1 & I_n & 1 \\
0 & I_n & 1 & I_n & 1
\end{bmatrix},
\]

where

\[
\begin{align*}
A & = \begin{bmatrix}
1 & \cdots & 1 \\
\vdots & \ddots & \vdots \\
1 & \cdots & 1
\end{bmatrix} \\
I_n & = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{bmatrix}. 
\end{align*}
\]
This matrix can clearly be constructed in polynomial time. We claim that \( C \in G_{n(n+4)} \) if and only if there is a \( T \in G_n \) such that \( AT = B \).

For ease of reference during the proof, we shall label the blocks of the matrix (and other matrices of the same size) as follows:

\[
\begin{bmatrix}
  n^2 & n & n & n & n \\
  a1 & b1 & c1 & d1 & e1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  a5 & b5 & c5 & d5 & e5 \\
\end{bmatrix}
\]

We shall also use \( a, b, c, d \) and \( e \) to refer to the sets of indices \( \{1, \ldots, n^2\}, \{n^2 + 1, \ldots, n^2 + n\}, \{n^2 + n + 1, \ldots, n^2 + 2n\}, \{n^2 + 2n + 1, \ldots, n^2 + 3n\} \) and \( \{n^2 + 3n + 1, \ldots, n^2 + 4n\} \) respectively, so that the columns indexed by these sets correspond to the blocks described above. We define \( a_i = i \) for \( i \leq n^2 \), and \( b_i = n^2 + i, c_i = n^2 + n + i, d_i = n^2 + 2n + i \) and \( e_i = n^2 + 3n + i \) for \( i \leq n \) so that, for example, the \( i \)th column indexed by \( c \) is column \( c_i \).

We first show that if \( AT = B \) for some \( T \in G_n \) then \( C \in G_{n(n+4)} \), by showing how to write \( C \) as a product of phone call matrices. Let

\[
Y_1 = \begin{bmatrix}
  n^2 & n & 3n \\
  1 & 4 & 0 \\
  1 & 1 & 0 \\
  0 & 0 & I_{3n} \\
\end{bmatrix}, \quad Y_T = \begin{bmatrix}
  n(n+3) & n \\
  I_{n(n+3)} & 0 \\
  0 & T \\
\end{bmatrix}
\]
\[
Y_2 = \prod_{i=1}^{n} C[d_i, e_i], \quad Y_3 = \prod_{i=1}^{n} C[c_i, d_i], \quad Y_4 = C[\{c_1, \ldots, c_n\}],
\]
\[
Y_5 = \prod_{i=1}^{n} C[b_i, e_i], \quad Y_6 = \prod_{i=1}^{n} C[c_i, e_i].
\]

Since \( n \geq 2 \) it follows from Lemma 6.13 that \( Y_1 \) is a product of phone call matrices in \( G_{n(n+4)} \). The assumption that \( T \in G_n \) means that \( Y_7 \) can be written as a product of phone call matrices in \( G_{n(n+4)} \). Matrices \( Y_2 \) through to \( Y_6 \) are all explicitly defined as products of phone call matrices or conference call matrices, and we note that the products in \( Y_2, Y_3, Y_5 \) and \( Y_6 \) can be taken in any order since their terms commute. It is then straightforward to check that \( C = Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 \).

Conversely, assume that \( C \in G_{n(n+4)} \) and fix a sequence of phone call matrices \( C_1, \ldots, C_q \) such that \( C = C_1 \cdots C_q \). Without loss of generality we may clearly assume that \( AC_1 \cdots C_t < AC_1 \cdots C_t C_{t+1} \) for all \( t \). By defining a particular scattered subsequence of these factors, we shall construct an element \( T \in G_n \) satisfying \( AT = B \).

We start by observing that for any scattered subsequence \( C_{t_1}, \ldots, C_{t_p} \) of \( C_1, \ldots, C_q \) we must have \( C_{t_1} \cdots C_{t_p} \preceq C \). We proceed by establishing a series of claims.

**Claim 1.** For each \( t \), if \( C_t \) is of the form \( C[i, j] \) with \( i \in a \cup b \) and \( j \in c \cup d \cup e \), then \( C_t = C[b_k, e_k] \) for some \( k \in \{1, \ldots, n\} \). Moreover, for each \( k \), the matrix \( C[b_k, e_k] \) must appear exactly once in the sequence \( C_1, \ldots, C_q \).

*Proof of claim:* Suppose that \( C_t = C[i, j] \), where \( i \in a \cup b \) and \( j \in c \cup d \cup e \). Since \( C_t \preceq C \) we note that \( i \notin a \), otherwise \( C \) would contain a 1 in block \( a_3 \), \( a_4 \) or \( a_5 \). Thus \( i = b_k \) for some \( k \leq n \). Similarly, we note that \( j \notin c \cup d \), since otherwise \( C \) would contain a 1 in block \( b_3 \) or \( d_2 \). Finally, note that if \( j = e_l \) for some \( l \neq k \) then \( C \) would contain an off-diagonal 1 in block \( b_5 \).
It now follows that the only way that $C$ can contain the identity matrix in block $b5$ is for each of the matrices $C[b_k, e_k]$ to occur at least once in the sequence $C_1, \ldots, C_q$. Assume for contradiction that $C[b_k, e_k]$ occurs more than once in the sequence, and suppose that $C_r$ and $C_s$ are the first two occurrences of this matrix. If $C_t$ is a factor occurring between $C_r$ and $C_s$ then $C_r C_t C_s \preceq C$. We cannot have $C_t$ of the form $C[b_k, j]$ with $j \in a \cup b$, as otherwise $C_r C_t C_s$ — and therefore also $C$ — would have a 1 in block $a5$ or an off-diagonal 1 in block $b5$. Similarly, it cannot be of the form $C[e_k, j]$ with $j \in c \cup d \cup e$, since then $C_r C_t C_s$ would have a 1 in block $b3$ or $d2$ or an off-diagonal 1 in block $b5$. By the first part of the claim, $C_t$ cannot be of the form $C[b_k, e_k]$ as it lies strictly between the first two factors of this form. Therefore $C_t$ is not of the form $C[b_k, j]$ or $C[e_k, j]$ for any $j$, so $C_r$ commutes with $C_t$. Since $C_t$ was an arbitrary factor between $C_r$ and $C_s$, and since $C_r = C_s$ is an idempotent, we have $C_r C_{r+1} \cdots C_{s-1} C_s = C_r C_s C_{r+1} \cdots C_{s-1} = C_r C_{r+1} \cdots C_{s-1}$, contradicting our assumption that the sequence $C_1, \ldots, C_q$ contains no redundant factors and thus proving the claim.

For $k \in \{1, \ldots, n\}$ let $w_k$ be the unique index such that $C_{w_k} = C[b_k, e_k]$.

**Claim 2.** For each $k \leq n$, column $k$ of block $e1$ in the product $C_1 \cdots C_{w_k-1}$ is equal to the zero vector, but the same column in $C_1 \cdots C_{w_k}$ is equal to column $k$ of $A$.

**Proof of claim:** We first show that in the product $C_1 \cdots C_{w_k-1}$, the $k$th column of block $e1$ is equal to the zero vector. Assume that the vector is non-zero, so there is some $C_s$ with $s < w_k$ such that $C_1 \cdots C_{s-1}$ has the zero vector in this location, but $C_1 \cdots C_s$ does not. The factor $C_s$ must be of the form $C[e_k, j]$ and by Claim 1 and the fact that $s < w_k$, we have $j \in c \cup d \cup e$. We note that $j \notin c$, otherwise $C_s C_{w_k}$ — and hence also $C$ — would have a 1 in block $b3$. Similarly we note that $j \notin e$,
otherwise $C_sC_{w_k}$ would have an off-diagonal 1 in block $b5$. This means $C_s$ must be of the form $C[e_k, d_i]$, but then columns $e_k$ and $d_i$ of $C_1 \cdots C_s$ are identical to each other, so this product has a 1 in block $d1$, contradicting $C_1 \cdots C_s \preceq C$. Thus the $k$th column of block $e1$ in the product $C_1 \cdots C_{w_k-1}$ is equal to the zero vector.

Since the last element of the product $C_1 \cdots C_{w_k}$ is $C_{w_k} = C[b_k, e_k]$, columns $b_k$ and $e_k$ of the product are identical and the second part of the claim will follow if the $k$th column of block $b1$ in this product is equal to the $k$th column of $A$. Suppose this does not hold; since it does hold in $C$, the sequence $C_{w_k+1}, \ldots, C_q$ must include a subsequence which transforms the $k$th column of $b1$ into the $k$th column of $A$. Since this subsequence modifies column $b_k$, at least one element of the subsequence must be of the form $C[b_k, j]$. Notice that $j \notin a$, since otherwise $C_{w_k}C[b_k, j] \preceq C$ would have a 1 in block $a5$. Similarly, $j \notin b$, since otherwise $C_{w_k}C[b_k, j]$ would have an off-diagonal 1 in block $b5$. The first part of Claim 1 now shows that the only remaining possibility is $j = e_k$. However, this would result in two factors of $C$ both equal to $C[b_k, e_k]$, contradicting the second part of Claim 1. Hence column $k$ of block $b1$ in the product $C_1 \cdots C_{w_k}$ is equal to the $k$th column of $A$, completing the proof of the claim. 

To provide a convenient starting point for induction, let $C_0$ be the $n(n+4) \times n(n+4)$ identity matrix, so that $C = C_0C_1 \cdots C_q$.

**Claim 3.** For all $0 \leq t \leq q$, every column in block $c1$, $d1$ or $e1$ of $C_0C_1 \cdots C_t$ is equal to the sum of some (possibly empty) collection of columns of $A$.

**Proof of claim:** We prove this by induction on $t$. When $t = 0$ this is clearly true, as each of these columns is equal to $0$, the sum of an empty collection of columns. For $t \geq 1$ we shall assume that the condition holds for the columns of the appropriate blocks in $C_1 \cdots C_{t-1}$. By Claim 1, $C_t$ is of one of the following forms:

(i) $C[i, j]$ with $i, j \in a \cup b$,
(ii) $C[i,j]$ with $i, j \in c \cup d \cup e$, \\
(iii) $C[b_k,e_k]$ for some $k$.

In case (i) the columns of blocks $c1, d1$ and $e1$ of $C_1 \cdots C_t$ are the same as the corresponding columns in the product $C_1 \cdots C_{t-1}$, which satisfy the condition by assumption. In case (ii) two of the columns of blocks $c1, d1$ and $e1$ of $C_1 \cdots C_t$ are equal to the sum of the corresponding columns from the product $C_1 \cdots C_{t-1}$, and so they satisfy the condition of the claim, and the rest of the columns are the same as the corresponding columns from $C_1 \cdots C_{t-1}$. In case (iii) by Claim 2, column $k$ of block $e1$ of $C_1 \cdots C_t$ is equal to the $k$th column of $A$, so satisfies the claim, and all other columns of blocks $c1, d1$ and $e1$ are equal to the corresponding columns in $C_1 \cdots C_{t-1}$. The claim follows by induction. ■

To prove the theorem we want to construct a matrix $T \in G_n$ such that $AT = B$. Consider those phone call matrices $C_t$ (in order) for which there exists $k \leq n$ such that $w_k < t$ and column $k$ of block $e1$ in the product $C_1 \cdots C_t$ is strictly larger than the same column in the product $C_1 \cdots C_{t-1}$. (Intuitively, these are the factors which modify some column $k$ of block $e1$ subsequent to the factor $C_{w_k} = C[b_k,e_k]$ which copies column $k$ of $A$ into this column.) Let $C_{t_1}, \ldots, C_{t_p}$ denote the subsequence of these matrices. We shall assume that this sequence is non-empty, since otherwise $B = A$ and $AT = B$ is trivially satisfied by $T = I_n$.

**Claim 4.** Each element of the sequence $C_{t_1}, \ldots, C_{t_p}$ is of the form $C_{t_r} = C[e_j,e_k]$ for some $j, k \leq n$ such that $w_j < t_r$ and $w_k < t_r$.

**Proof of claim:** Let $C_{t_r}$ be an element of this subsequence. From the definition of the subsequence there is some $k \leq n$ such that $w_k < t_r$ and column $k$ of block $e1$ of $C_1 \cdots C_{t_r}$ is strictly larger than the same column in $C_1 \cdots C_{t_r-1}$. Thus $C_{t_r}$ must be of the form $C[i,e_k]$ for some $i \leq n(n+4)$. Claim 1 tells us that $i \notin a$. Since
\( t_r > w_k \), Claim 1 also tells us that \( i \notin b \). By monotonicity, all columns in block \( d1 \) of \( C_1 \cdots C_{t_r} \) are 0, so \( i \notin d \). Assume for contradiction that \( C_{t_r} = C[e_j, e_k] \) for some \( j \).

We observe several relations between columns of various matrices:

- column \( j \) of \( A \) is equal to column \( j \) of block \( c1 \) in \( C \),
- by monotonicity, column \( j \) of block \( c1 \) in \( C \) is greater than or equal to the same column in \( C_1 \cdots C_{t_r} \),
- since the last factor in the product \( C_1 \cdots C_{t_r} \) is \( C_{t_r} = C[e_j, e_k] \), columns \( c_j \) and \( e_k \) are equal in the resulting matrix,
- by our choice of \( k \), column \( k \) of block \( e1 \) of \( C_1 \cdots C_{t_r} \) is strictly greater than the same column in \( C_1 \cdots C_{t_r-1} \),
- since \( w_k < t_r \), by Claim 2 and monotonicity we know that column \( k \) of block \( e1 \) in \( C_1 \cdots C_{t_r-1} \) is greater than or equal to column \( k \) of \( A \).

Putting these together, we find that column \( j \) of \( A \) is strictly greater than column \( k \) of \( A \), which contradicts the maximal column condition. Thus \( C_{t_r} \) must be of the form \( C[e_j, e_k] \) for some \( j \).

We know that \( w_k < t_r \) by definition of the sequence, so for the second part of the claim it is enough to observe that \( t_r \neq w_j \) since \( C_{t_r} \neq C[b_j, e_j] \), and if \( t_r < w_j \) then we would have \( C[e_j, e_k]C[b_j, e_j] \preceq C \) and so \( C \) would contain an off-diagonal 1 in block \( b5 \), which is not the case.

We see from this claim that there are only two types of phone call matrix in the sequence \( C_1, \ldots, C_q \) which can modify a column of block \( e1 \). The first matrix in the sequence which modifies column \( i \) of block \( e1 \) is \( C_{w_i} \), and all subsequent matrices (if any) which modify this column are elements of the subsequence \( C_{t_1}, \ldots, C_{t_r} \).
We now define a new sequence of phone call matrices in $G_n$. For $1 \leq r \leq p$, if $C_{t_r} = C[e_i, e_j]$ then let $D_r = C[i, j] \in G_n$.

**Claim 5.** For each $r \leq p$ and $k \leq n$, column $k$ of block $e_1$ of $C_1 \cdots C_{t_r}$ is equal to the zero vector if $t_r < w_k$, or equal to column $k$ of $AD_1 \cdots D_r$ otherwise.

**Proof of claim:** By Claim 2, for each $k \leq n$ the $k$th column of block $e_1$ in the product $C_1 \cdots C_{w_k-1}$ is equal to the zero vector. By monotonicity, if $t_r < w_k$ then the same column is equal to the zero vector in the product $C_1 \cdots C_{t_r}$. We prove by induction on $r$ that for each $k$ such that $w_k < t_r$, column $k$ of block $e_1$ of $C_1 \cdots C_{t_r}$ is equal to column $k$ of $AD_1 \cdots D_r$.

Let $t_0 = 0$. Then $t_0 < w_k$ for all $k$ so the claim holds for $r = 0$. Now assume the claim holds for $r - 1$. That is, for each $k$ such that $w_k < t_{(r-1)}$, column $k$ of block $e_1$ of $C_1 \cdots C_{t_{(r-1)}}$ is equal to column $k$ of $AD_1 \cdots D_{(r-1)}$. Given $k \leq n$ such that $w_k < t_r$ we first consider the differences between column $k$ of block $e_1$ in $C_1 \cdots C_{t_{(r-1)}}$ and the same column in $C_1 \cdots C_{t_r-1}$. No elements of the subsequence $C_{t_1}, \ldots, C_{t_p}$ occur among the matrices $C_{t_{(r-1)}+1}, \ldots, C_{t_r-1}$ and we have noted that the only other factor of $C$ which can possibly modify column $k$ of block $e_1$ is $C_{w_k}$. Thus the two columns under consideration are identical unless $t_{(r-1)} < w_k < t_r$, and in this case, by Claim 2, column $k$ of block $e_1$ in $C_1 \cdots C_{t_r-1}$ is equal to column $k$ of $A$. If $t_{(r-1)} < w_k < t_r$ then, since $C_{w_k}$ comes earlier in the sequence than any other matrix which can modify column $k$ of block $e_1$, we see that $C_{t_r}$ is the first element of the sequence $C_{t_1}, \ldots, C_{t_p}$ which can possibly modify column $k$ of block $e_1$, and thus $D_r$ is the first element of the sequence $D_1, \ldots, D_p$ which can possibly modify column $k$. Thus column $k$ of $AD_1 \cdots D_{r-1}$ is equal to column $k$ of $A$, and is therefore equal to column $k$ of $e_1$ in $C_1 \cdots C_{t_{r-1}}$. If $t_{(r-1)} < w_k < t_r$ does not hold then, since we are only considering $k$ such that $w_k < t_r$, we must have $w_k < t_{(r-1)}$. Thus by the inductive hypothesis we know that column $k$ of block $e_1$ of $C_1 \cdots C_{t_{(r-1)}}$ is equal to column $k$ of $AD_1 \cdots D_{r-1}$. Since none of the matrices
Therefore no element of the sequence \( C_{t(r-1)+1}, \ldots, C_{t_r-1} \) modify this column, column \( k \) of \( e_1 \) in \( C_1 \cdots C_{t_r-1} \) is again equal to column \( k \) of \( AD_1 \cdots D_{r-1} \). Thus for all \( k \) such that \( w_k < t_r \), regardless of whether or not \( t_{(r-1)} < w_k < t_r \), we know that column \( k \) of \( e_1 \) in \( C_1 \cdots C_{t_r-1} \) is equal to column \( k \) of \( AD_1 \cdots D_{r-1} \).

By Claim 4, \( C_{t_r} = C[e_i, e_j] \) for some \( i, j \leq n \) such that \( w_i, w_j < t_r \), and then \( D_r \) is defined to be \( C[i, j] \). Let \( k \leq n \) be such that \( w_k < t_r \). We first consider the case when \( k \notin \{i, j\} \). Since \( k \notin \{i, j\} \), column \( k \) of block \( e_1 \) of \( C_1 \cdots C_{t_r} \) is equal to the same column in \( C_1 \cdots C_{t_r-1} \). By the previous paragraph, this column is equal to column \( k \) of \( AD_1 \cdots D_{r-1} \), and again because \( k \notin \{i, j\} \), this is equal to column \( k \) of \( AD_1 \cdots D_r \). In the case where \( k \in \{i, j\} \), column \( k \) of block \( e_1 \) of \( C_1 \cdots C_{t_r} \) is equal to the sum of columns \( i \) and \( j \) of block \( e_1 \) of \( C_1 \cdots C_{t_r-1} \). Since \( w_i, w_j < t_r \), by the previous paragraph this is equal to the sum of columns \( i \) and \( j \) of \( AD_1 \cdots D_{r-1} \). Since \( D_r = C[i, j] \), column \( k \) of \( AD_1 \cdots D_r \) is also equal to the sum of columns \( i \) and \( j \) of \( AD_1 \cdots D_{r-1} \), which completes the proof of the claim. 

We now define \( T = D_1 \cdots D_p \). For each \( k \leq n \), the only factors of \( C \) which can possibly modify column \( k \) of block \( e_1 \) are \( C_{w_k} \) and elements of the subsequence \( C_{t_1}, \ldots, C_{t_p} \). Therefore column \( k \) of block \( e_1 \) of \( C \) is equal to the same column in \( C_1 \cdots C_{t_p} \) if \( w_k \leq t_p \), or it is equal to the same column in \( C_1 \cdots C_{w_k} \) if \( t_p < w_k \).

If \( w_k \leq t_p \) then by Claim 5 column \( k \) of \( e_1 \) in \( C_1 \cdots C_{t_p} \) is equal to column \( k \) of \( AT = AD_1 \cdots D_p \), and therefore column \( k \) of \( e_1 \) in \( C \) is also equal to column \( k \) of \( AT \).

If \( t_p < w_k \) then \( C_{w_k} \) is the last factor of \( C \) to modify column \( k \) of block \( e_1 \), so by Claim 2, column \( k \) of \( e_1 \) in \( C \) is equal to column \( k \) of \( A \). We know from Claim 4 that if \( C_{t_r} = C[e_j, e_k] \) for some \( j \) and \( r \) then \( w_k < t_r \). However \( t_r \leq w_k \) for all \( r \leq p \), so no element of the sequence \( C_{t_1}, \ldots, C_{t_p} \) is equal to \( C[e_j, e_k] \) for any \( j \leq n \). Therefore no element of the sequence \( D_1, \ldots, D_p \) is equal to \( C[j, k] \) for any \( j \leq n \), and so column \( k \) of \( AT = AD_1 \cdots D_p \) is also equal to column \( k \) of \( A \).
In both cases, column $k$ of block $e_1$ of $C$ is equal to column $k$ of $AT$. Therefore $AT$ is equal to block $e_1$ of $C$, which is equal to $B$, and so $T \in G_n$ satisfies $AT = B$ as required to complete the proof. 

**Theorem 6.16**

*The decision problem GJP is NP-complete.*

**Proof.** Given $X,Y \in G_n$, GJP asks if $X \leq_J Y$, or in other words if there exist $U,V \in G_n$ such that $UYV = X$.

By Theorem 6.5, each element of $G_n$ can be written as a product of no more than $n(n - 1)/2$ phone call matrices. Thus if there exist $U,V \in G_n$ such that $UYV = X$ a non-deterministic Turing machine can guess and verify this in polynomial time. This shows that GJP is in NP. We show GJP is NP-hard via a polynomial time reduction from MGTP.

Let $A, B \in \mathbb{B}_n$ such that $A$ satisfies the maximal column condition (although we will not make use of the maximal column condition in this proof). We will construct matrices $X,Y \in G_{2n(2n+1)}$ such that $UYV = X$ for some $U,V \in G_{2n(2n+1)}$ if and only if $AT = B$ for some $T \in G_n$.

Let $X$ and $Y$ be the $2n(2n + 1) \times 2n(2n + 1)$ gossip matrices

$$X = \begin{bmatrix} (2n)^2 & n & n \\ & 1 & B \ I_n \\ & 0 & 1 \\ 1 & 1 & \end{bmatrix}^{n \times n}, \quad Y = \begin{bmatrix} (2n)^2 & n & n \\ & 1 & A \ I_n \\ & 0 & 1 \\ 1 & 1 & \end{bmatrix}^{n \times n}.$$

These can clearly be constructed in polynomial time, and by Lemma 6.13 we know that $X,Y \in G_{2n(2n+1)}$.

If there exists $T \in G_n$ such that $AT = B$ then let $U = I_{2n(2n+1)}$ and let $V$ be the $2n(2n + 1) \times 2n(2n + 1)$ gossip matrix.
A simple calculation shows that $UYV = X$.

Conversely, if there exist $U, V \in G_{2n(2n+1)}$ such that $UYV = X$, then we consider what structure the matrices $U$ and $V$ could possibly have by considering them as products of phone call matrices.

Recall that left multiplication of $Y$ by a phone call matrix $C[i, j]$ replaces rows $i$ and $j$ of $Y$ with their sum, whilst right multiplication by $C[i, j]$ replaces columns $i$ and $j$ of $Y$ with their sum. Thus, by regarding $U$ and $V$ as products of phone call matrices, we see that $X$ can be built from $Y$ by successively replacing either two rows or two columns with their sum.

The last $(2n)^2$ rows of $Y$ all contain 1s in each of the last $2n$ columns, whilst the first $2n$ rows of $X$ each have a 0 in one of the last $2n$ columns. It follows that $U$ cannot contain $C[i, j]$ as a factor if $i \leq 2n$ and $j > 2n$. Each of the rows $n+1, \ldots, 2n$ of $Y$ contain 1s in each of the last $n$ columns, whilst the first $n$ rows of $X$ each have a 0 in one of the last $n$ columns. Thus $U$ cannot contain $C[i, j]$ as a factor if $i \leq n$ and $n < j \leq 2n$. The factors of $U$ are therefore phone call matrices $C[i, j]$ such that $i$ and $j$ are either both less than $n$, both between $n$ and $2n$, or both greater than $2n$. Both $X$ and $Y$ contain the $n \times n$ identity matrix in the top right corner. It follows that $U$ cannot contain $C[i, j]$ as a factor if $i, j \leq n$ are distinct, as this would result in an off-diagonal zero in this $n \times n$ submatrix of $X$. Therefore, for some $D \in G_n, E \in G_{(2n)^2}$ we must have
CHAPTER 6. GOSSIP MONOIDS

\[
U = \begin{bmatrix}
I_n & 0 & (2n)^2 \\
0 & D & 0 \\
0 & E & 0 \\
m & m & (2n)^2
\end{bmatrix}
\]

It is easy to check that \( UY = Y \). Thus \( YV = UYV = X \).

Consider now the action of \( V \) by right multiplication on \( Y \). The first \((2n)^2\) columns of \( Y \) all contain 1s in each of the first \(2n\) rows, whilst the last \(2n\) columns of \( X \) each have a 0 in one of the first \(2n\) rows, and so \( V \) cannot contain \( C[i,j] \) as a factor if \( i \leq (2n)^2 \) and \( j > (2n)^2 \). Each of the last \(n\) columns of \( Y \) contains a 1 in row \( n+1 \), whilst columns \((2n)^2 + 1, \ldots, (2n)^2 + n\) of \( X \) each contain a 0 in row \( n+1 \), and so \( V \) cannot contain \( C[i,j] \) as a factor if \( (2n)^2 < i \leq (2n)^2 + n \) and \( j > (2n)^2 + n \). It now follows that there must be some \( F \in G_{(2n)^2} \) and \( T,G \in G_n \) such that

\[
V = \begin{bmatrix}
(2n)^2 & m & m \\
F & 0 & 0 \\
0 & T & G \\
m & m & (2n)^2
\end{bmatrix}
\]

and then \( YV = X \) tells us that \( AT = B \) as required.

**Theorem 6.17**

The decision problems GLP and GRP are NP-complete.

**Proof.** Given \( X, Y \in G_n \), GLP asks if \( X \leq_L Y \) and GRP asks if \( X \leq_R Y \).

Since \( X \leq_L Y \iff X^T \leq_R Y^T \) and \( G_n^T = G_n \) it suffices to show that GRP is NP-complete. By Theorem 6.5, each element of \( G_n \) can be written as a product of
no more than \(n(n - 1)/2\) phone call matrices. Thus if there exists \(V \in G_n\) such that \(YV = X\) a non-deterministic Turing machine can guess and verify this in polynomial time. This shows that GRP is in NP.

We show GRP is NP-hard using the same reduction as in the previous theorem. We showed that if there exists \(T \in G_n\) such that \(AT = B\) then there are \(U, V \in G_{2n(2n+1)}\), with \(U = I_{2n(2n+1)}\) such that \(UYV = X\). Clearly then we also have \(YV = X\). Conversely, if there exists \(V \in G_{2n(2n+1)}\) such that \(YV = X\) then by taking \(U = I_{2n(2n+1)}\) we see there are \(U, V \in G_{2n(2n+1)}\) such that \(UYV = X\) and so, by the above proof, there exists \(T \in G_n\) such that \(AT = B\). \(\square\)
Chapter 7

One-Directional Gossip Monoids

In the next class of monoids we consider the situation where instead of phone calls the gossips communicate by messages which transfer information in just one direction. This variation on gossip does not seem to have been studied as widely as regular gossip, although some authors have considered similar variations. For example Harary and Raghavachari [17] consider the situation where each gossip can send an email to multiple other gossips at once but certain pairs of gossips are unable to directly communicate with each other. However, the main result of the paper, which concerns the smallest number of emails required to spread all scandals to all gossips, doesn’t apply to our situation where each message has only one recipient.

The same problem applied to our situation is essentially the one-directional analogue of the gossip problem. This is considered in [18] where it is shown by a simple proof that $2n - 2$ is the smallest number of messages which can be used to spread all scandals to all gossips.
Definition 7.1: Message Matrix
Given $a, b \leq n$, let $M[a, b] \in \mathbb{B}_n$ be the matrix defined by

$$m[a, b]_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } (i, j) = (a, b), \\
0 & \text{otherwise.}
\end{cases}$$

We call matrices of this form message matrices.

Example 7.2
In $\mathbb{B}_5$ the message matrix $M[3, 5]$ is

$$M[3, 5] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$ 

All message matrices have this general form - they have 1s on the diagonal and only one off-diagonal 1.

Definition 7.3: One-Directional Gossip Monoid
The one-directional gossip monoid of rank $n$, $\bar{G}_n$, is the monoid generated by all message matrices of order $n$:

$$\bar{G}_n = \langle \{M[a, b] : a, b \leq n\} \rangle.$$ 

In the one-directional gossip monoids the multiplication simplifies considerably: right multiplying a matrix $A \in \mathbb{B}_n$ by the message matrix $M[a, b]$ has the effect of replacing column $b$ of $A$ with the sum of columns $a$ and $b$, while leaving all other columns unchanged, and left multiplying $A$ by $M[a, b]$ has the effect of replacing row $a$ of $A$ with the sum of rows $a$ and $b$, while leaving all other rows unchanged.

If $i \neq l$ and $j \neq k$ then $M[i, j]$ and $M[k, l]$ commute. In general if $A$ and $B$ are disjoint sets then all message matrices of the form $M[a, b]$ with $a \in A$ and $b \in B$ commute with each other.
Example 7.2 (continued)

Let $A \in B_5$ be as in the previous example:

$$A = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}.$$

Then the product $AM[3, 5]$ is the result of adding column 3 of $A$ to column 5 while leaving the other columns unchanged:

$$AM[3, 5] = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}.$$

This matrix represents the state of knowledge obtained by a message from gossip 3 to 5 after starting at the state represented by $A$.

The product $M[3, 5]A$ is the result of adding row 5 of $A$ to row 3 while leaving the other rows unchanged:

$$M[3, 5]A = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}.$$

The result of Brouwer, Draisma and Frenk [4] concerning the maximum length of a non-redundant product of calls, Theorem 6.5, is much simpler in the one-directional case:

**Theorem 7.4**

The maximum length of a product of messages, such that somebody learns something new in each message, is $n(n - 1)$. 
Proof. For each $1 \leq i \leq n-1$ let
\[ R_i = M[i, i+1]M[i, i+2] \cdots M[i, n], \]
and for each $2 \leq i \leq n$ let
\[ L_i = M[i, i-1]M[i, i-2] \cdots M[i, 1]. \]

Then the product
\[ R_{n-1}R_{n-2} \cdots R_1 L_2 L_3 \cdots L_n \]
contains $n(n-1)$ messages, and teaches every scandal to every gossip in such a way that one piece of information is learned in each message. Since only one piece of information is learned in each message this product clearly has maximum length. \hfill \Box

As promised in the introduction, we now prove that $U_n \subseteq \bar{G}_n$.

**Theorem 7.5**

For each $n$, the unitriangular monoid of rank $n$ is a submonoid of the one-directional gossip monoid of rank $n$.

Proof. Given $A \in U_n$ define, for each $i \leq n$, the set $S_i = \{ j \leq n : j > i \text{ and } a_{i,j} = 1 \}$, and let $A_i$ be the matrix
\[ A_i = \prod_{j \in S_i} M[i, j], \]
where the product can be taken in any order since all message matrices in the product commute, and we take the empty product to be the identity matrix. We claim that $A = A_n A_{n-1} \cdots A_1$, which is sufficient to show that $A \in \bar{G}_n$. We show this by showing, by induction on $k$, that the last $k$ rows of $A_n A_{n-1} \cdots A_{n+1-k}$ are equal to the last $k$ rows of $A$. 


Since the definition of $S_i$ contains the condition $j > i$ the set $S_n$ is empty. Thus $A_n$ is the identity matrix, the last row of which is the same as the last row of any element of $U_n$. This proves the base case.

We assume that the last $k - 1$ rows of $A_n A_{n-1} \cdots A_{n+2-k}$ are equal to the last $k - 1$ rows of $A$. Since the message matrices in the product $A_n A_{n-1} \cdots A_{n+2-k}$ are all of the form $M[i,j]$ with $j > i > n + 1 - k$ we see that the first $n + 1 - k$ rows and $n + 2 - k$ columns of this matrix are all equal to the corresponding rows and columns in the identity matrix. In particular column $n + 1 - k$ has a 1 only in row $n + 1 - k$, so multiplying on the right by the message matrix $M[n + 1 - k, j]$ adds a 1 to the $(n + 1 - k, j)$ position but does not add any other 1s. The result of multiplying on the right by $A_{n+1-k}$ is therefore to add a 1 in the location of every off-diagonal 1 in row $n + 1 - k$ of $A$. Row $n + 1 - k$ of $A_n A_{n-1} \cdots A_{n+2-k}$ is equal to the corresponding row of the identity matrix, so the same row in $A_n A_{n-1} \cdots A_{n+1-k}$ is equal to the corresponding row of $A$. By the inductive hypothesis the last $k$ rows of $A_n A_{n-1} \cdots A_{n+2-k}$ are equal to the last $k - 1$ rows of $A$, and multiplying on the right by $A_{n+1-k}$ only affects row $n + 1 - k$. Therefore the last $k$ rows of $A_n A_{n-1} \cdots A_{n+1-k}$ are equal to the last $k$ rows of $A$.

We will show in the next proof that the idempotents of the one-directional gossip monoids turn out to be precisely those elements which are reflexive and transitive when interpreted as relations. Note that these are precisely the same as the idempotents of the reflexive monoids. Therefore $\bar{G}_n$ is the idempotent generated submonoid of $R_n$.

**Theorem 7.6**

The idempotents of $\bar{G}_n$, the one-directional gossip monoid of rank $n$, are the matrices $A \in \mathbb{B}_n$ which satisfy $a_{i,j} = 1 \iff i \sim j$ for some reflexive and transitive relation $\sim$ on the set $\{1, \ldots, n\}$. 

Proof. Let $A \in \bar{G}_n$ be a one-directional gossip matrix. Then $A = \prod_{r=0}^{m} M[a_r, b_r]$ for some message matrices $M[a_0, b_0], \ldots, M[a_m, b_m]$.

Let $A$ be idempotent and define $\sim_A$ by $i \sim_A j \iff a_{i,j} = 1$. Since $a_{i,i} = 1$ for all $i$, $\sim_A$ is reflexive. Since $A = A^2$ we have

$$a_{i,j} = \sum_{k=1}^{n} a_{i,k}a_{k,j},$$

so

$$i \sim_A k \text{ and } k \sim_A j \Rightarrow a_{i,k} = a_{k,j} = 1$$

$$\Rightarrow a_{i,j} = 1$$

$$\Rightarrow i \sim_A j,$$

and $\sim_A$ is transitive.

This shows that if $A$ is an idempotent of the one-directional gossip monoid of rank $n$ then it satisfies $a_{i,j} = 1 \iff i \sim j$ for some reflexive and transitive relation $\sim$.

For the converse, let $\sim$ be a reflexive and transitive relation and define $A$ by $a_{i,j} = 1 \iff i \sim j$. We first show that $A \in \bar{G}_n$. If $\sim$ is equality then $A$ is the $n \times n$ identity matrix, which is an element of $\bar{G}_n$. Otherwise there is at least one pair $i \neq j$ such that $i \sim j$ and we will show that $A$ can be written as the product of all the message matrices $M[i,j]$ such that $i \sim j$, taken in any order.

Let $P$ be a product of all message matrices $M[i,j]$ such that $i \sim j$, in some order. For each $i,j$ such that $a_{i,j} = 1$ we have $M[i,j]_{i,j} = 1$, so $p_{i,j} = 1$ by monotonicity. Thus $A \preceq P$.

Next we show that if $B \preceq A$ and $C \preceq A$ then $BC \preceq A$. If $(BC)_{i,j} = 1$ then there is some $k$ such that $b_{i,k} = c_{k,j} = 1$, so $a_{i,k} = a_{k,j} = 1$ and so $i \sim k$ and $k \sim j$. 

By transitivity \( i \sim j \) so \( a_{i,j} = 1 \). This shows that \( BC \preceq A \). Since \( M[i,j] \preceq A \) for each \( i, j \) such that \( i \sim j \), by induction \( P \preceq A \), and so \( A = P \). Therefore \( A \in \bar{G}_n \).

By monotonicity we have \( A \preceq A^2 \) and by the same reasoning as in the previous paragraph, since \( A \preceq A \) we have \( A^2 \preceq A \). Thus \( A = A^2 \) and so \( A \) is idempotent.

Since the one-directional gossip monoids are defined by generators rather than by the structure of the matrix we consider the membership problem for these monoids too:

**Definition 7.7: One-Directional Gossip Membership Problem (GMP)**

*Given \( C \in \mathbb{B}_n \), the one-directional gossip membership problem asks if \( C \in \bar{G}_n \).*

We also define the following decision problem which will be used as an intermediate step in all polynomial time reductions in this chapter:

**Definition 7.8: One-Directional Gossip Transformation Problem (GTP)**

*Given \( A, B \in \mathbb{B}_n \), the one-directional gossip transformation problem asks if there exists \( T \in \bar{G}_n \) such that \( AT = B \).*

We will use the following lemma in our proofs that \( \bar{G} \text{T} \) and \( \bar{G} \text{M} \) are NP-hard:

**Lemma 7.9**

*Let \( A \in \mathbb{R}_n \) and \( T \in \bar{G}_n \), and let \( B = AT \). Write \( T \) as a product of message matrices \( T = M_1 \cdots M_q \). If \( i, j \leq n \) are such that \( a_{i,j} = 0 \) but \( b_{i,j} = 1 \) then there is some unique \( t \leq q \) such that position \( (i,j) \) of \( AM_1 \cdots M_{t-1} \) is equal to 0 but the same position in \( AM_1 \cdots M_t \) is equal to 1. Moreover, \( M_t \) is of the form \( M[l,j] \) for some \( l \neq j \) such that \( b_{i,l} = b_{i,j} = 1 \), and position \( (i,l) \) of \( AM_1 \cdots M_{t-1} \) is equal to 1.*

**Proof.** The first part of the lemma follows immediately from monotonicity. Multiplying a matrix \( X \) on the right by a message matrix \( M[a,b] \) results in a matrix which differs from \( X \) only on column \( b \). Since multiplying \( AM_1 \cdots M_{t-1} \) on the right
by $M_t$ results in a matrix which differs on column $j$, we must have $M_t = M[l, j]$ for some $l \leq n$.

Column $j$ of $AM_1 \cdots M_t$ is equal to the sum of columns $l$ and $j$ of $AM_1 \cdots M_{t-1}$. Since column $j$ of $AM_1 \cdots M_t$ has a 1 in row $i$, at least one of columns $l$ and $j$ of $AM_1 \cdots M_{t-1}$ must contain a 1 in row $i$. We know that column $j$ of $AM_1 \cdots M_{t-1}$ has a 0 in row $i$, so $AM_1 \cdots M_{t-1}$ has a 1 in position $(i, l)$. By monotonicity we have $b_{i,l} = 1$. Since position $(i, j)$ of $M_t$ is equal to 1, by monotonicity we also have $b_{l,j} = 1$.

Finally, since $M_t = M[l, j]$, column $j$ of $AM_1 \cdots M_t$ is equal to the sum of columns $l$ and $j$ of $AM_1 \cdots M_{t-1}$. Therefore, since position $(i, j)$ of $AM_1 \cdots M_t$ is equal to 1 but the same position is equal to 0 in $AM_1 \cdots M_{t-1}$, we must have that position $(i, l)$ is equal to 1 in $AM_1 \cdots M_{t-1}$. $\Box$

Note that this lemma can also be applied to a single matrix $C \in \bar{G}_n$ by letting $A = I_n$ and $T = C = M_1 \cdots M_q$. The lemma then says that if $i \neq j$ are such that $c_{i,j} = 1$ then there is some unique $t \leq q$ such that position $(i, j)$ of $M_1 \cdots M_{t-1}$ is equal to 0 but the same position in $M_1 \cdots M_t$ is equal to 1. Moreover, $M_t$ is of the form $M[l, j]$ for some $l \neq j$ such that $c_{i,l} = c_{l,j} = 1$, and position $(i, l)$ of $M_1 \cdots M_{t-1}$ is equal to 1.

**Theorem 7.10**

The decision problem $\bar{G}$TP is NP-hard.

**Proof.** Given $A, B \in \mathbb{B}_n$, $\bar{G}$TP asks if there exists $T \in \bar{G}_n$ such that $AT = B$. We show $\bar{G}$TP is NP-hard via a polynomial time reduction from DSP.

Let $H = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$, and let $k \in \mathbb{N}$ be such that $0 < k < n$. We will construct matrices $A, B \in \mathbb{B}_{4n+2}$ such that $AT = B$ for some $T \in \bar{G}_{4n+2}$ if and only if $H$ has a dominating set of size $k$.

Recall that we can regard $H$ as an $n \times n$ Boolean matrix. Let $A$ and $B$ be the $(4n + 2) \times (4n + 2)$ Boolean matrices

\[A = \begin{pmatrix} 
I_n & C \\
C^T & B 
\end{pmatrix}, \quad 
B = \begin{pmatrix} 
0_{n \times n} & B \\
B^T & I_n 
\end{pmatrix} \]
For reference, we shall label the blocks of $A$ and $B$ (and other matrices of the same size) as follows:
We shall also use $a, b, c, d, e,$ and $f$ to refer to the sets of indices $\{1, \ldots, n\}, \{n+1, \ldots, 2n\}, \{2n+1, \ldots, 3n\}, \{3n+1, \ldots, 4n\}, \{4n+1\}$ and $\{4n+2\}$ respectively, so that the columns indexed by these sets correspond to the blocks described above. We define $a_i = i, b_i = n+i, c_i = 2n+i$ and $d_i = 3n+i$ for $i \leq n$, and $e_1 = 4n+1$ and $f_1 = 4n+2$ so that, for example, the $i$th column indexed by $c$ is column $c_i$.

We have already seen that $H$ has a dominating set of size $k$ if and only if there is a vector $\mathbf{v} \in \mathbb{B}^n$ with $k$ 1s such that $(H + I_n)\mathbf{v} = \mathbf{1}$. If such a vector exists then there is a permutation $p$ of $\{1, 2, \ldots, n\}$ such that

$$v_i = \begin{cases} 
1 & \text{if } p(i) \leq k, \\
0 & \text{otherwise}.
\end{cases}$$

We show that if such a vector $\mathbf{v}$ exists then there is a $T \in \bar{G}_{4n+2}$ such that $AT = B$ by writing $T$ as a product of message matrices. Let
\[ T_1 = \prod_{i=1}^{n} M[a_i, d_{p-1(i)}], \quad T_2 = \prod_{i=1}^{n} M[c_i, a_i], \quad T_3 = \prod_{i=1}^{n} M[b_i, c_i], \]
\[ T_4 = \prod_{i=1}^{n} M[d_{p-1(i)}, b_i], \quad T_5 = \prod_{i=1}^{n} M[e_1, a_i], \quad T_6 = \prod_{i=1}^{k} M[d_{p-1(i)}, e_1], \]
\[ T_7 = \prod_{i=1}^{n} M[f_1, d_i]. \]

Note that the product in \( T_6 \) only goes up to \( i = k \) whereas each other product goes to \( i = n \). The multiplication can be taken in any order within each of these products since the terms commute.

For each \( i \leq 7 \) the matrix \( T_i \) is equal to the identity matrix except on one block. These exceptions are as follows:

- block d1 of \( T_1 \) is equal to the unique permutation matrix \( P \) such that \( Pe_i = e_{p(i)} \) for each \( i \leq n \), where \( e_i \) is the \( i \)th unit vector,
- block a3 of \( T_2 \) is equal to \( I_n \),
- block c2 of \( T_3 \) is equal to \( I_n \),
- block b4 of \( T_4 \) is equal to \( P^{-1} \),
- block a5 of \( T_5 \) is equal to \( 1^T \),
- block e4 of \( T_6 \) is equal to \( v \) since \( v_i = 1 \iff p(i) \leq k \),
- block d6 of \( T_7 \) is equal to \( 1^T \).

We define \( T = T_1 T_2 T_3 T_4 T_5 T_6 T_7 \in \tilde{G}_{4n+2} \). Now, by a simple but lengthy matrix multiplication we can verify that
and then, since \((H + I_n)v = 1\), we have \(AT = B\).

We now assume that there exists \(T \in G_{4n+2}\) such that \(AT = B\) and we will show that there exists a vector \(v \in \mathbb{B}^n\) with \(k\) 1s such that \((H + I_n)v = 1\). Note that \(T\) can be written as a product of message matrices, so we fix some such product \(T = M_1 \cdots M_q\). Without loss of generality we may clearly assume that \(M_1 \cdots M_t < M_1 \cdots M_t M_{t+1}\) for all \(t\).

Block \(b_1\) of \(A\) is equal to 0, but in \(B\) the same block is equal to \(I_n\). Since \(A \in R_n\) we can use Lemma 7.9. For each \(i \leq n\) we must have some \(t_i \leq q\) such that position \((a_i, b_i)\) is equal to 0 in \(AM_1 \cdots M_{t_i-1}\) and equal to 1 in \(AM_1 \cdots M_{t_i}\), and \(M_{t_i} = M[l, b_i]\) for some \(l \leq 4n + 2\) such that \(b_{a_i, l} = b_{l, b_i} = 1\) and position \((i, l)\) of \(AM_1 \cdots M_{t_i-1}\) is equal to 1. Row \(a_i\) and column \(b_i\) of \(B\) show that \(l\) is equal to either \(a_i\) or one of \(d_1, \ldots, d_n\). We show that \(l \neq a_i\) by the following considerations:

- \(b_{b_i, c_i} = 1\) but \(a_{b_i, c_i} = 0\). By Lemma 7.9 and row \(b_i\) of \(B\) we see that the message matrix \(M[b_i, c_i]\) occurs in the product.

- \(b_{c_i, a_i} = 1\) but \(a_{c_i, a_i} = 0\). By Lemma 7.9 and row \(c_i\) of \(B\) we see that the
message matrix \( M[c_i, a_i] \) occurs in the product.

- Since \( b_{c_i, b_i} = 0 \), the message matrix \( M[a_i, b_i] \), if it appears in the product, cannot occur after the message matrix \( M[c_i, a_i] \).

- Since \( b_{a_i, c_i} = 0 \), the message matrix \( M[a_i, b_i] \), if it appears in the product, cannot occur before the message matrix \( M[b_i, c_i] \).

- Since \( b_{b_i, a_i} = 0 \), the message matrix \( M[c_i, a_i] \) must occur earlier in the product than the message matrix \( M[b_i, c_i] \). Therefore each message matrix in the product is either after \( M[c_i, a_i] \) or before \( M[b_i, c_i] \), so the message matrix \( M[a_i, b_i] \) cannot occur in the product.

Thus \( l = d_{j_i} \) for some \( j_i \leq n \).

Define a function \( p : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\} \) by letting \( p(i) = j_i \) for each \( i \leq n \).

We wish to show that \( p \) is a permutation, which will follow if \( p \) is injective. Assume for contradiction that it is not injective. Then there is some \( i \neq j \) such that \( p(i) = p(j) \), and we will let \( l = d_{p(i)} = d_{p(j)} \). Without loss of generality we assume that \( M_{t_i} = M[l, b_i] \) occurs earlier in the product than \( M_{t_j} = M[l, b_j] \). Our first use of Lemma 7.9 told us that position \( (a_i, l) \) of \( AM_1 \cdots M_{t_i-1} \) is equal to 1, and so by monotonicity, the same position is equal to 1 in \( AM_1 \cdots M_{t_j-1} \). Then, since \( M_{t_j} = M[l, b_j] \), position \( (a_i, b_j) \) is equal to 1 in \( AM_1 \cdots M_{t_j} \), and by monotonicity we have \( b_{a_i, b_j} = 1 \), a contradiction. Therefore \( p \) is a permutation.

For each \( i \leq n \) we have established that the message matrix \( M_{t_i} = M[d_{p(i)}, b_i] \) occurs in the product and that position \( (a_i, d_{p(i)}) \) is equal to 1 in \( AM_1 \cdots M_{t_i-1} \). The same position is equal to 0 in \( A \), so by Lemma 7.9 there must be some \( s_i < t_i \) such that position \( (a_i, d_{p(i)}) \) is equal to 0 in \( AM_1 \cdots M_{s_i-1} \) and equal to 1 in \( AM_1 \cdots M_{s_i} \), and \( M_{s_i} = M[l, d_{p(i)}] \) for some \( l \leq 4n + 2 \) such that \( b_{a_i, l} = b_{l, d_{p(i)}} = 1 \) and position
(a_i, l) of AM_1 \cdots M_{s_i - 1} is equal to 1. By looking at row a_i and column d_{p(i)} of B we see that l is equal to either a_i, one of d_1, \ldots, d_n or f_1.

We cannot have l = f_1, otherwise the matrices M[l, d_{p(i)}] and M[d_{p(i)}, b_i] would result in a 1 in block b6. We cannot have l = d_{p(j)} for any j \neq i, otherwise either:

- s_i < t_j in which case, since position (a_i, l) is equal to 1 in AM_1 \cdots M_{s_i - 1} and M_{t_j} = M[d_{p(j)}, b_j] = M[l, b_j] occurs later in the product, we would get an off-diagonal 1 in block b1 of B,

- t_j < s_i < t_i in which case, since position (a_j, l) is equal to 1 in AM_1 \cdots M_{t_j - 1} and M_{s_i} = M[d_{p(j)}, d_{p(i)}] = M[l, d_{p(i)}] and M_{t_i} = M[d_{p(i)}, b_i] occur later in the product, in that order, we would again get an off-diagonal 1 in block b1 of B.

The only remaining possibility is l = a_i, so the message matrix M_{s_i} = M[a_i, d_{p(i)}] occurs in the product for each i \leq n.

For i \leq n we have b_{e_1,a_i} = 1 but a_{e_1,a_i} = 0. Thus by Lemma 7.9 there is some r_i such that M_{r_i} = M[l, a_i] for some l \neq a_i such that b_{e_1,l} = b_{l,a_i} = 1. Looking at row e_1 and column a_i of B we see that l = e_1. We must have r_i > s_i, otherwise the product would have M[e_1, a_i] before M[a_i, d_{p(i)}] and we would have a 1 in block d5 of B.

We now consider, for each i \leq n, whether M[d_{p(i)}, e_1] could be in the product. It cannot come before M_{s_i} = M[e_1, a_i], otherwise we would get a 1 in block d1 of B. Thus if M[d_{p(i)}, e_1] does occur in the product it does so after M_{s_i}, and therefore after M_{r_i} = M[a_i, d_{p(i)}]. We would then have b_{a_i,e_1} = 1, which is only the case if i \leq k. Thus M[d_{p(i)}, e_1] can only occur in the product if i \leq k.

For j \leq n we have b_{d_j,e_1} = 1 but a_{d_j,e_1} = 0. Thus by Lemma 7.9 there is some q_j such that M_{q_j} = M[l, e_1] for some l \neq e_1 such that b_{d_j,l} = b_{l,e_1} = 1 and position (d_j, l) is equal to 1 in AM_1 \cdots M_{q_j - 1}. Looking at row d_j and column e_1 of B we
see that \( l = d_i \) for some \( i \). We define a function \( f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\} \) such that for each \( j \leq n \), the message matrix \( M_{d_j} \) is equal to \( M[d_{f(j)}, e_1] \). We know that \( M[d_{p(i)}, e_1] \) can only occur in the product if \( i \leq k \), so the image of \( p^{-1}f \) is contained in \( \{1, 2, \ldots, k\} \).

Block \( d_4 \) is equal to \( H + I_n \) in both \( A \) and \( B \), so by monotonicity it is equal to \( H + I_n \) in \( AM_1 \cdots M_{d_j} \) for each \( j \). For each \( j \) we know that position \( (d_j, d_{f(j)}) \) of \( AM_1 \cdots M_{d_j-1} \) is equal to 1, so position \( (j, f(j)) \) of \( H + I_n \) equals 1. Therefore the vector \( u \in B^n \) defined by

\[
u_i = \begin{cases} 1 & \text{if } i = f(j) \text{ for some } j, \\ 0 & \text{otherwise,} \end{cases}
\]

satisfies \( (H + I_n)u = 1 \). Any vector \( v \in B^n \) such that \( v \succeq u \) satisfies \( (H + I_n)v \succeq (H + I_n)u = 1 \), so \( (H + I_n)v = 1 \). Since the image of \( p^{-1}f \) is contained in \( \{1, 2, \ldots, k\} \), the image of \( f \) is contained in \( \{p(1), p(2), \ldots, p(k)\} \) and so the vector \( u \) has at most \( k \) 1s. Thus there is at least one \( v \in B^n \) with exactly \( k \) 1s such that \( v \succeq u \), and so \( H \) has a dominating set of size \( k \). \(\square\)

**Theorem 7.11**

The decision problem \( \bar{GMP} \) is NP-complete.

**Proof.** Given \( C \in B_n \), \( \bar{GMP} \) asks if \( C \in \bar{G}_n \).

By Theorem 7.4, each element of \( \bar{G}_n \) can be written as a product of no more than \( n(n-1) \) phone call matrices. Thus a non-deterministic Turing machine can guess an element of \( \bar{G}_n \) in polynomial time and check if it is equal to \( C \). This shows that \( \bar{GMP} \) is in NP. We show \( \bar{GMP} \) is NP-hard via a polynomial time reduction from \( \bar{GTP} \).

Let \( A, B \in B_{n-1} \). We will define a Boolean matrix \( C \in B_{4n} \) which can be constructed in polynomial time, such that \( C \in \bar{G}_{4n} \) if and only if \( AT = B \) for some \( T \in \bar{G}_{n-1} \).
Let $A'$ and $B'$ be the $n \times n$ Boolean matrices
\[
A' = \begin{bmatrix}
\cdots & 1 \\
A & 0 \\
0 & \cdots \\
\end{bmatrix} \quad \text{and} \quad B' = \begin{bmatrix}
\cdots & 1 \\
B & 0 \\
0 & \cdots \\
\end{bmatrix}.
\]

Then let $C$ be the $4n \times 4n$ Boolean matrix
\[
C = \begin{bmatrix}
I_n & A' & B' & 0 \\
1 & I_n & 0 & 0 \\
0 & I_n & 1 & 1 \\
0 & 0 & I_n & I_n \\
\end{bmatrix} \in \mathbb{B}_{4n}.
\]

For reference, we shall label the blocks of $C$ (and other matrices of the same size) as follows:
\[
\begin{bmatrix}
a_1 & b_1 & c_1 & d_1 \\
a_2 & b_2 & c_2 & d_2 \\
a_3 & b_3 & c_3 & d_3 \\
a_4 & b_4 & c_4 & d_4 \\
\end{bmatrix} \in \mathbb{B}_{4n}.
\]

We shall also use $a$, $b$, $c$ and $d$ to refer to the sets of indices $\{1, \ldots, n\}$, $\{n + 1, \ldots, 2n\}$, $\{2n + 1, \ldots, 3n\}$ and $\{3n + 1, \ldots, 4n\}$ respectively, so that the columns indexed by these sets correspond to the blocks described above. We define $a_i = i$, $b_i = n + i$, $c_i = 2n + i$ and $d_i = 3n + i$ for $i \leq n$ so that, for example, the $i$th column indexed by $c$ is column $c_i$.

If there is a $T \in \tilde{G}_{n-1}$ such that $AT = B$ then we show that $C \in \tilde{G}_{4n}$ by writing it as a product of message matrices. Let
$Y_1 = \prod_{i=1}^{n} \prod_{j=1}^{n} M[c_i, d_j]$, $Y_2 = \prod_{i,j:a_i,j=1} M[a_j, c_i]$, $Y_3 = \prod_{i=1}^{n} M[b_n, a_i]$

$Y_4 = M[a_n, b_n]$, $Y_5 = \prod_{i=1}^{n-1} \prod_{j=1}^{n} M[b_i, a_j]$, $Y_6 = \prod_{i=1}^{n} M[c_i, b_i]$

$Y_7 = \begin{bmatrix}
I_n & 0 & 0 & 0 \\
0 & I_n & 0 & 0 \\
0 & 0 & T & 0 \\
0 & 0 & 0 & I_n
\end{bmatrix}$

$Y_8 = \prod_{i=1}^{n} M[d_i, c_i]$

The assumption that $T \in \bar{G}_{n-1}$ means that $Y_7$ can be written as a product of message matrices in $\tilde{G}_{4n}$. The other matrices are all explicitly defined as products of message matrices, and we note that the products in $Y_1, Y_2, Y_3, Y_5, Y_6$ and $Y_8$ can be taken in any order since their terms commute. It is then straightforward to check that $C = Y_1Y_2Y_3Y_4Y_5Y_6Y_7Y_8$.

Now we assume that $C \in \tilde{G}_{4n}$ and show that there exists $T \in \bar{G}_{n-1}$ such that $AT = B$. By this assumption, $C$ can be written as a product of message matrices in $\tilde{G}_{4n}$. Fix some such product $C = M_1 \cdots M_q$. Without loss of generality we may clearly assume that $M_1 \cdots M_t \prec M_1 \cdots M_t M_{t+1}$ for all $t$.

Since block $a2$ of $C$ is equal to 1, for each $i, j \leq n$ Lemma 7.9 tells us that the message matrix $M[l, a_j]$ occurs in the product for some $l \neq a_j$ such that
Looking at row $b_i$ and column $a_i$ of $C$ shows that $l = b_i$. Thus the message matrix $M[b_i, a_j]$ occurs in the product.

Since block $b_3$ of $C$ is equal to $I_n$, the same lemma tells us that for each $i \leq n$ there is some $r_i \leq q$ such that $M_{r_i} = M[l, b_i]$ for some $l \neq b_i$ such that $c_{r_i,l} = c_{l,b_i} = 1$. Looking at row $c_i$ and column $b_i$ of $C$ shows that $l = c_i$. Thus $M_{r_i} = M[c_i, b_i]$. Since block $a_3$ of $C$ is equal to 0, for $i, j \leq n$ the message matrix $M[b_i, a_j]$ must occur before $M[c_i, b_i]$.

Since block $c_1$ of $C$ is equal to $B'$, for each $i, j \leq n$ such that $b'_{i,j} = 1$ Lemma 7.9 tells us there is a message matrix $M_{s_{i,j}}$ in the product such that position $(a_i, c_j)$ is equal to 0 in $M_1 \cdots M_{s_{i,j} - 1}$ but the same position is equal to 1 in $M_1 \cdots M_{s_{i,j}}$, and that $M_{s_{i,j}} = M[l, c_j]$ for some $l \neq c_j$ such that $a_{i,l} = a_{l,c_j} = 1$. Looking at row $a_i$ and column $c_j$ of $C$ shows that $l$ is equal to either $a_i$ or $c_k$ for some $k \leq n$. This accounts for all of the 1s in block $c_1$, so there is no other matrix $M_s$ such that products $M_1 \cdots M_{s-1}$ and $M_1 \cdots M_s$ differ on block $c_1$.

Let $i, j, k, l \leq n$ with $i \neq j$ and $k \neq l$. We already know that both $M[b_i, a_i]$ and $M[c_i, b_i]$ occur in the product, in that order. If the message matrix $M[a_i, c_j]$ is in the product then, since block $c_2$ of $C$ is equal to 0, it must occur before $M[b_i, a_i]$. If the message matrix $M[c_k, c_l]$ is in the product then, since block $b_3$ of $C$ is equal to $I_n$, it must occur after $M[c_1, b_l]$, which occurs after $M[b_i, a_i]$. Thus, all message matrices of the form $M[a_i, c_j]$ occur before any message matrices of the form $M[c_k, c_l]$, and we can choose some $t \leq q$ such that for each $i, j \leq n$ if $M_{s_{i,j}} = M[a_i, c_j]$ then $s_{i,j} \leq t$ and if $M_{s_{i,j}} = M[c_k, c_l]$ then $s_{i,j} > t$.

We let $\hat{A} \in B_n$ be equal to block $c_1$ of $M_1 \cdots M_t$. The differences between this matrix and the same block in $C = M_1 \cdots M_q$ (which is equal to $B'$) are due to the matrices of the form $M[c_k, c_l]$. Thus if we define $\hat{T} \in G_n$ as the product of the message matrices $M[k, l] \in G_n$ such that $M[c_k, c_l]$ occurs in the product $M_1 \cdots M_q$ (with the multiplication in the same order) then we have $\hat{A}\hat{T} = B'$. It follows that
\( \hat{A} \preceq B' \), so \( \hat{A} \) has no 1s in row \( n \) or column \( n \).

Recall that for each \( i \leq n \) we defined \( r_i \) such that \( M_{r_i} = M[c_i, b_i] \).

**Claim.** For each \( j \leq n \), column \( j \) of block \( c_1 \) of \( M_{r_1} \cdots M_{r_j} \) is equal to column \( j \) of \( \hat{A} \).

**Proof of claim:** Let \( j \leq n \). Since \( \hat{A} \preceq B' \) there are three possibilities for each \( i \leq n \):

- \( \hat{a}_{i,j} = b'_{i,j} = 0 \),
- \( \hat{a}_{i,j} = 0 \) but \( b'_{i,j} = 1 \),
- \( \hat{a}_{i,j} = b'_{i,j} = 1 \).

In the first case, since \( b'_{i,j} = 0 \), position \( (a_i, c_j) \) is equal to 0 in \( C \), so by monotonicity it is also equal to 0 in \( M_{r_1} \cdots M_{r_j} \). Since \( \hat{a}_{i,j} = 0 \) and \( i \) was arbitrary, column \( j \) of block \( c_1 \) of \( M_{r_1} \cdots M_{r_j} \) is equal to column \( j \) of \( \hat{A} \).

In the second case position \( (a_i, c_j) \) is equal to 0 in \( M_{r_1} \cdots M_{s_i,j} \), since \( \hat{A} \) is defined as block \( c_1 \) of this product. Since \( b'_{i,j} = 1 \) there is some matrix \( M_{s_{i,j}} \) in the product equal to either \( M[a_i, c_j] \) or \( M[c_k, c_j] \) for some \( k \leq n \) such that position \( (a_i, c_j) \) is equal to 1 in \( M_{r_1} \cdots M_{s_{i,j}} \). By monotonicity we must have \( s_{i,j} > t \) and so \( M_{s_{i,j}} = M[c_k, c_j] \) for some \( k \). Since block \( b_3 \) of \( C \) has no off-diagonal 1s, \( M_{r_j} = M[c_j, b_j] \) must occur before \( M[c_k, c_j] \), so we have \( r_j < s_{i,j} \). We also know that position \( (a_i, c_j) \) is equal to 0 in \( M_{r_1} \cdots M_{r_{s_{i,j}-1}} \), so by monotonicity we know that the same position is equal to 0 in \( M_{r_1} \cdots M_{r_{s_{i,j}}-1} \). Since \( \hat{a}_{i,j} = 0 \) and \( i \) was arbitrary, column \( j \) of block \( c_1 \) of \( M_{r_1} \cdots M_{r_{s_{i,j}}-1} \) is equal to column \( j \) of \( \hat{A} \).

In the third case position \( (a_i, c_j) \) is equal to 1 in \( M_{r_1} \cdots M_{t} \), since \( \hat{A} \) is defined as block \( c_1 \) of this product. Since \( b'_{i,j} = 1 \) there is some matrix \( M_{s_{i,j}} \) in the product equal to either \( M[a_i, c_j] \) or \( M[c_k, c_j] \) for some \( k \leq n \) such that position \( (a_i, c_j) \) is equal to 0 in \( M_{r_1} \cdots M_{s_{i,j}-1} \). By monotonicity we must have \( s_{i,j} \leq t \) and so
\( M_{s_{i,j}} = M[a_i, c_j] \). We have already established that \( M[b_j, a_i] \) must occur in the product. Since block a3 of \( C \) is equal to 0, \( M_{r_j} = M[c_j, b_j] \) must occur after \( M[b_j, a_i] \), and since block c2 of \( C \) is equal to 0, \( M[a_i, c_j] \) must occur before \( M[b_j, a_i] \). Thus \( M_{r_j} = M[c_j, b_j] \) occurs after \( M[a_i, c_j] \), so we have \( r_j > s_{i,j} \). We also know that position \((a_i, c_j)\) is equal to 1 in \( M_1 \cdots M_{s_{i,j}} \), so by monotonicity we know that the same position is equal to 1 in \( M_1 \cdots M_{r_j} \). Since \( \hat{a}_{i,j} = 1 \) and \( i \) was arbitrary, column \( j \) of block c1 of \( M_1 \cdots M_{r_j} \) is equal to column \( j \) of \( \hat{A} \).

Block \( b_1 \) of \( C \) is equal to \( A' \), and \( A'_{n,n} = 1 \), so by Lemma 7.9 the message matrix \( M[l, b_n] \) must occur in the product for some \( l \neq b_n \) such that \( c_{a_n,l} = c_{l,b_n} = 1 \). Looking at row \( a_n \) and column \( b_n \) of \( C \) shows that \( l = a_n \). Thus \( M[a_n, b_n] \) occurs in the product.

We will show that if \( j \leq n - 1 \) then any message matrix of the form \( M[k, b_j] \) is equal to \( M[c_j, b_j] \). Since block b4 of \( C \) is equal to 0, \( k \) cannot be equal to \( d_i \) for any \( i \). Since blocks b2 and b3 of \( C \) are both equal to \( I_n \), \( k \) cannot be equal to \( b_i \) or \( c_i \) for any \( i \neq j \). Since \( A'_{n,j} = 0 \), \( k \) cannot be equal to \( a_n \). Thus \( k \) is equal to either \( c_j \) or \( a_i \) for some \( i \leq n - 1 \).

Assume for contradiction that \( k = a_i \) for some \( i \leq n - 1 \). We already know that the message matrices \( M[a_n, b_n], M[b_j, a_n] \) and \( M[b_n, a_i] \) occur at some point. Because there are no off-diagonal zeros in blocks a1 or b2 of \( C \), the matrix \( M[a_i, b_j] \) must occur before \( M[b_n, a_i] \), which must occur before \( M[a_n, b_n] \), which must occur before \( M[b_j, a_n] \). Thus \( M[a_i, b_j] \) occurs before \( M[b_j, a_n] \), but this would result in an off-diagonal zero in block a1 of \( C \), giving the required contradiction.

Thus the only message matrices in the product which can result in 1s in block b1 are \( M[a_n, b_n] \), which places a 1 in position \((a_n, b_n)\) but nowhere else in that block, and \( M_{r_j} = M[c_j, b_j] \) for each \( j \leq n \). Let \( j \leq n - 1 \). By the claim, column \( j \) of block c1 of \( M_1 \cdots M_{r_j} \) is equal to column \( j \) of \( \hat{A} \), so column \( j \) of block b1 of \( M_1 \cdots M_{r_j} \) must also be equal to column \( j \) of \( \hat{A} \). No other matrices in the product
modify column $j$ of block $b_1$, so column $j$ of $\hat{A}$ is equal to column $j$ of block $b_1$ in $C$, which is equal to column $j$ of $A'$. We already know that $\hat{A}$ has no 1s in column $n$. Thus $\hat{A}$ is equal to $A'$ except that $\hat{a}_{n,n} = 0$:

$$\hat{A} = \begin{bmatrix}
\begin{array}{c}
n-1 \\
A \\
0 \\
\hline
0 \\
1
\end{array}
\end{bmatrix}
\begin{array}{c}
n-1 \\
1
\end{array}.$$  

We know there is some $\hat{T} \in \bar{G}_n$ such that $\hat{A}\hat{T} = B'$. We now write $\hat{T}$ as a product of message matrices $T = T_1 \cdots T_p$ and consider any matrices in this product of the form $M[i,n]$ or $M[n,i]$ for some $i \leq n - 1$.

If $T_a = M[i,n]$ for some $a \leq p$ then the only possible differences between matrices $\hat{A}T_1 \cdots T_{a-1}$ and $\hat{A}T_1 \cdots T_a$ are on column $n$. By monotonicity we have $\hat{A} \preceq \hat{A}T_1 \cdots T_{a-1} \preceq \hat{A}T_1 \cdots T_a \preceq B'$ so column $n$ is equal to 0 in all of these matrices. Thus the message matrix $T_a$ is redundant in this product.

If $T_a = M[n,i]$ for some $a \leq p$ then column $i$ of $\hat{A}T_1 \cdots T_a$ is equal to the sum of columns $i$ and $n$ in $\hat{A}T_1 \cdots T_{a-1}$. Again we have $\hat{A} \preceq \hat{A}T_1 \cdots T_{a-1} \preceq \hat{A}T_1 \cdots T_a \preceq B'$ so column $n$ is equal to 0 in all of these matrices, so in $\hat{A}T_1 \cdots T_{a-1}$ the sum of columns $i$ and $n$ is simply equal to column $i$. Thus the message matrix $T_a$ is redundant in this product.

In either case $T_a$ is redundant in the product $\hat{A}T_1 \cdots T_p$, so we can remove any message matrix involving $n$ from the product $T_1 \cdots T_p$, and the resulting product gives us a matrix of the form

$$T' = \begin{bmatrix}
\begin{array}{c}
n-1 \\
T \\
0 \\
\hline
0 \\
1
\end{array}
\end{bmatrix}
\begin{array}{c}
n-1 \\
1
\end{array},$$  

for some $T \in \bar{G}_{n-1}$. Since only redundant factors were removed, this matrix also
satisfies \( \hat{AT}' = B' \), and comparing the top-left blocks of \( \hat{AT}' \) and \( B' \) gives us \( AT = B \) as required.

**Theorem 7.12**

The decision problem \( \bar{GJP} \) is NP-complete.

**Proof.** Given \( X,Y \in \bar{G}_n \), \( \bar{GJP} \) asks if \( X \leq_d Y \), or in other words if there exist \( U,V \in \bar{G}_n \) such that \( UYV = X \).

By Theorem 7.4, each element of \( \bar{G}_n \) can be written as a product of no more than \( n(n-1) \) message matrices. Thus if there exist \( U,V \in \bar{G}_n \) such that \( UYV = X \) a non-deterministic Turing machine can guess and verify this in polynomial time. This shows that \( \bar{GJP} \) is in NP. We show \( \bar{GJP} \) is NP-hard via a polynomial time reduction from \( \bar{GTP} \).

Let \( A,B \in \mathbb{B}_n \). We will construct matrices \( X,Y \in \bar{G}_{3n} \) such that \( UYV = X \) for some \( U,V \in \bar{G}_{3n} \) if and only if \( AT = B \) for some \( T \in \bar{G}_n \).

Let \( X \) and \( Y \) be the \( 3n \times 3n \) matrices

\[
X = \begin{bmatrix}
I_n & 0 & B \\
I_n & I_n & 0 \\
1 & 1 & 1
\end{bmatrix}^n \quad Y = \begin{bmatrix}
I_n & 0 & A \\
I_n & I_n & 0 \\
1 & 1 & 1
\end{bmatrix}^n
\]

First we show that \( X,Y \in \bar{G}_{3n} \). Given any \( M \in \mathbb{B}_n \), let

\[
M_1 = \prod_{i=1}^{n} \prod_{j=1}^{n} M[2n + i, 2n + j], \quad M_2 = \prod_{i=1}^{n} M[2n + i, n + i],
\]

\[
M_3 = \prod_{\{(i,j):m_{i,j}=1\}} M[i, 2n + j], \quad M_4 = \prod_{i=1}^{n} M[n + i, i].
\]

The multiplication can be taken in any order within each of these products since the terms commute. Multiplying these together results in the matrix
CHAPTER 7. ONE-DIRECTIONAL GOSSIP MONOIDS

\[
M_1M_2M_3M_4 = \begin{bmatrix}
I_n & 0 & M \\
\hline
I_n & I_n & 0 \\
1 & 1 & 1
\end{bmatrix}
\]

and so any matrix of this form is in \( \bar{G}_{3n} \).

Now we assume that there exists \( T \in \bar{G}_n \) such that \( AT = B \) and we show that there are \( U, V \in \bar{G}_{3n} \) such that \( UYV = X \). Let \( U = I_{3n} \) and let \( V \) be the \( 3n \times 3n \) one-directional gossip matrix

\[
V = \begin{bmatrix}
I_n & 0 & 0 \\
\hline
0 & I_n & 0 \\
0 & 0 & T
\end{bmatrix}
\]

Then \( UYV = X \).

Conversely, if there exist \( U, V \in \bar{G}_{3n} \) such that \( UYV = X \) then we consider the possible structure of \( U \) and \( V \) by considering them as products of message matrices.

The effect of multiplying a matrix on the left by the message matrix \( M[i, j] \) is to replace row \( i \) with the sum of rows \( i \) and \( j \). Since the last \( 2n \) rows of \( X \) are equal to the last \( 2n \) rows of \( Y \), any factors of \( U \) which are equal to \( M[i, j] \) with \( i > n \) are redundant.

Since the last \( 2n \) rows of \( Y \) each contain a 1 in at least one of the columns \( n+1, \ldots, 2n \), but each of the first \( n \) rows of \( X \) contains a 0 in each of these columns, there are no factors of \( U \) of the form \( M[i, j] \) with \( i \leq n \) and \( j > n \). Since the
top-left block of both $X$ and $Y$ is $I_n$, we see that no factor of $U$ is equal to $M[i, j]$ with distinct $i, j \leq n$.

Thus every factor of $U$ is redundant, so $UY = Y$ and therefore $YV = X$.

The effect of multiplying a matrix on the right by the message matrix $M[i, j]$ is to replace column $j$ with the sum of columns $i$ and $j$. Since the first $2n$ columns of $X$ are equal to the first $2n$ columns of $Y$, any factors of $V$ which are equal to $M[i, j]$ with $j \leq 2n$ are redundant.

Since the first $2n$ columns of $Y$ each contain a 1 in one of the rows $n + 1, \ldots, 2n$, but each of the last $n$ columns of $X$ contains a 0 in each of these rows, there are no factors of $V$ of the form $M[i, j]$ with $i \leq 2n$ and $j > 2n$. Thus the only irredundant factors of $V$ are of the form $M[i, j]$ with $i, j > 2n$. Therefore multiplying $Y$ on the right by $V$ has the same effect as multiplying on the right by

$$V' = \begin{pmatrix}
I_n & 0 & 0 \\
0 & I_n & 0 \\
0 & 0 & T
\end{pmatrix}^n,$$

for some $T \in \bar{G}_n$. Now $YV' = YV = X$ and this gives us $AT = B$ as required. 

**Theorem 7.13**

The decision problems $\bar{GLP}$ and $\bar{GRP}$ are NP-complete.

**Proof.** Given $X, Y \in \bar{G}_n$, $\bar{GLP}$ asks if $X \leq_L Y$ and $\bar{GRP}$ asks if $X \leq_R Y$.

Since $X \leq_L Y \iff X^T \leq_R Y^T$ and $\bar{G}_n = \bar{G}_n^T$ it suffices to show that $\bar{GRP}$ is NP-complete. By Theorem 7.4, each element of $\bar{G}_n$ can be written as a product of no more than $n(n - 1)$ message matrices. Thus if there exists $V \in \bar{G}_n$ such that $YV = X$ a non-deterministic Turing machine can guess and verify this in polynomial time. This shows that $\bar{GRP}$ is in NP.
We show $\overline{\text{GRP}}$ is NP-hard using the same reduction as in the previous theorem. We showed that if there exists $T \in \overline{G}_n$ such that $AT = B$ then there are $U, V \in \overline{G}_{3n}$, with $U = I_{3n}$ such that $UYV = X$, and so we have $YV = X$. Conversely, if there is $V \in \overline{G}_{3n}$ such that $YV = X$ then by taking $U = I_{3n}$ we see there are $U, V \in \overline{G}_{3n}$ such that $UYV = X$ and then by the above proof, there exists $T \in \overline{G}_n$ such that $AT = B$. \qed
Chapter 8

Catalan Monoids

The next class of monoids we consider are isomorphic to the monoids of non-increasing order-preserving transformations (we shall define precisely what this means soon, as well as describing the isomorphism). These monoids of transformations are typically known as Catalan monoids, with $C_n$ denoting the monoid of non-increasing order-preserving transformations on \{1, 2, \ldots, n\}. The name of the Catalan monoids is justified in [19], where Higgins shows that the number of elements in $C_n$ is equal to the $n$th Catalan number, $\frac{1}{n+1} \binom{2n}{n}$.

Some authors refer to these monoids as monoids of non-decreasing parking functions, denoted $NDPF_n$, such as Hivert and Thiéry, who provide a presentation of the monoids in [20].

In this thesis we shall stray from convention slightly by using $C_n$ to denote the monoid of $n \times n$ Boolean matrices isomorphic to $NDPF_n$ in which we are interested. When writing transformations on \{1, 2, \ldots, n\} we shall compose functions from left to right and write the operand on the left. The reason for this is explained later.

As the name suggests, a non-decreasing parking function is an example of a type of function known as a parking function. The name “parking function” comes from a situation described by Konheim and Weiss in [23]. Consider a street with $n$
parking spaces and $n$ cars driving down it, each with a passenger in the passenger’s seat. Each passenger is asleep (perhaps tired from spending the last two chapters gossiping) but wakes up just before reaching one of the parking spaces and asks the driver to park. The driver parks in the first available space, without reversing, or leaves the street if no spaces are available. If the passenger in the $i$th car wakes up just before the $a_i$th parking space, then the parking functions are precisely the maps $i \mapsto a_i$ which allow all $n$ cars to park on the street.

The “non-decreasing” part of the name refers to the property

$$i \leq j \implies [i] \phi \leq [j] \phi.$$  

A function satisfying this property is also known as order-preserving, and this is the name we shall use for such a function in this thesis. Confusingly, the term non-decreasing can also be used to describe a function satisfying the property

$$i \leq [i] \phi$$

for all $i$. Similarly, a function is said to be non-increasing if it satisfies

$$[i] \phi \leq i$$

for all $i$. We use the latter definition of a non-decreasing function in this thesis, so to avoid confusion we shall refer to $NDPF_n$ as the monoid of order-preserving parking functions and denote it $O^-_n$. (We use a $-$ symbol to contrast with another monoid, $O^+_n$, which we will soon define.)

If $\phi$ is an order-preserving parking function then each passenger travels at least as far as the previous passenger before waking up, and so each car parks further down the road than the previous car. Thus, for each $i \leq n$, car $i$ parks in spot $i$. We must therefore have $[i] \phi \leq i$ for each $i$, and so $\phi$ is non-increasing. Conversely, it is easy to see that if $\phi$ is order-preserving and non-increasing then it is a parking function. Therefore $O^-_n$ is the monoid of non-increasing order-preserving transformations on $\{1, 2, \ldots, n\}$, and we shall use this as our formal definition.
**Definition 8.1: Monoid of Order-Preserving Parking Functions**

Given a transformation $\phi$ on $\{1, 2, \ldots, n\}$ we say that $\phi$ is non-increasing if $[i]\phi \leq i$ for all $i$, and we say that $\phi$ is order-preserving if $[i]\phi \leq [j]\phi$ for all $i \leq j$. We call the set of all non-increasing order-preserving transformations on $\{1, 2, \ldots, n\}$ the monoid of order-preserving parking functions on $\{1, 2, \ldots, n\}$ and denote it $O_n^-$. 

We can also consider a very similar situation, where the $n$ cars approach the street from the opposite direction so that they pass parking space $n$ first, and car $n$ approaches first, followed by car $n - 1$, and so on. If the map $i \mapsto a_i$ allows all cars to park in this situation then we shall call it a reverse parking function. It is not hard to see that the order-preserving reverse parking functions are precisely the non-decreasing order-preserving transformations.

**Definition 8.2: Monoid of Order-Preserving Reverse Parking Functions**

Given a transformation $\phi$ on $\{1, 2, \ldots, n\}$ we say that $\phi$ is non-decreasing if $i \leq [i]\phi$ for all $i$, and we say that $\phi$ is order-preserving if $[i]\phi \leq [j]\phi$ for all $i \leq j$. We call the set of all non-decreasing order-preserving transformations on $\{1, 2, \ldots, n\}$ the monoid of order-preserving reverse parking functions on $\{1, 2, \ldots, n\}$ and denote it $O_n^+$. 

These two monoids are isomorphic to each other by, for example, the map which sends a transformation $\phi \in O_n^-$ to the transformation $\psi \in O_n^+$ such that $[i]\psi = n + 1 - [n + 1 - i]\phi$. Our primary interest is in $O_n^+$ and a particular monoid of Boolean matrices isomorphic to it, although any results we prove about $O_n^+$ also apply to $O_n^-$, and we shall make use of both $O_n^+$ and $O_n^-$ in the next chapter when we introduce the double Catalan monoids.

We can visually represent a transformation $\phi$ on $\{1, 2, \ldots, n\}$ as follows: write the numbers $1, 2, \ldots, n$ in a row, then again in another row directly below the first, then for each $i$ draw a line from $i$ in the top row to $[i]\phi$ in the bottom row.
Example 8.3

\[ \phi \]

\[ \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]

is the function

\[ \begin{align*}
\end{align*} \]

The condition that elements of \( O^+_n \) are non-decreasing means that none of the lines go to the left, and the condition that elements of \( O^+_n \) are order-preserving means that the lines do not cross. We see that none of the lines in the above example go to the left or cross each other, so \( \phi \in O^+_n \). Given two transformations \( \phi \) and \( \psi \) we can write \( \phi \) directly above \( \psi \) in this way and follow the lines from top to bottom to find the composition \( \phi \psi \).

Example 8.3 (continued)

\[ \psi \]

\[ \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]

is the function

\[ \phi \psi \]

\[ \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]
Since this thesis concerns monoids of Boolean matrices, we now define two monoids of Boolean matrices and prove they are isomorphic to $O_n^+$ and $O_n^-$. 

Consider, for each $1 \leq k \leq n - 1$, the $n \times n$ matrix $\delta_k$ defined by

$$(\delta_k)_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ or } (i,j) = (k,k+1), \\ 0 & \text{otherwise}, \end{cases}$$

and the $n \times n$ matrix $\hat{\delta}_k$ defined by

$$(\hat{\delta}_k)_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ or } (i,j) = (k+1,k), \\ 0 & \text{otherwise}. \end{cases}$$

These are message matrices: $\delta_k = M[k,k+1]$ and $\hat{\delta}_k = M[k+1,k]$. The monoids defined by these generators are therefore submonoids of $\bar{G}_n$.

**Definition 8.4: Catalan Monoids of rank $n$**

The Catalan monoid of rank $n$, denoted $C_n$, is the monoid generated by the matrices $\delta_k$ for $1 \leq k \leq n - 1$:

$$C_n = \langle \{\delta_k : 1 \leq k \leq n - 1\} \rangle.$$

In some contexts we shall refer to this as the upper Catalan monoid of rank $n$ and denote it $C_n^+$. We also define the lower Catalan monoid of rank $n$, denoted $C_n^-$, to be the monoid generated by the matrices $\hat{\delta}_k$ for $1 \leq k \leq n - 1$:

$$C_n^- = \langle \{\hat{\delta}_k : 1 \leq k \leq n - 1\} \rangle.$$

Since $C_n^+, C_n^- \subseteq \bar{G}_n$, the multiplication simplifies in the same way: right multiplying a matrix $A \in \mathbb{B}_n$ by the matrix $\delta_k$ has the effect of replacing column $k+1$ of $A$ with the sum of columns $k$ and $k+1$, while leaving all other columns unchanged. Right multiplying a matrix $A \in \mathbb{B}_n$ by the matrix $\hat{\delta}_k$ has the effect of replacing column $k$ of $A$ with the sum of columns $k$ and $k+1$, while leaving all other columns unchanged.
It is not hard to see that these monoids are isomorphic to each other by the map which sends $A \in C_n^+ \rightarrow A^R \in C_n^-$, the result of rotating $A$ by 180 degrees. Since the generators of $C_n$ are all upper unitriangular, $C_n$ is a submonoid of $U_n$.

We define a map $F : O_n^+ \rightarrow C_n^+$, which we will soon show is an isomorphism, by letting $F(\phi)$ be the matrix

$$f(\phi)_{i,j} = \begin{cases} 1 & \text{if } i \leq j \leq [i] \phi, \\ 0 & \text{otherwise.} \end{cases}$$

We show $F(\phi)$ is an element of $C_n^+$ by writing it as a product of generators. For each $i \leq n$ let $A_i = \delta_i \delta_{i+1} \cdots \delta_{[i] \phi - 1}$ (we regard the empty product as $I_n$). Then we write $A = A_n A_{n-1} \cdots A_1$.

Given $i \leq n$, for each $i \leq j \leq [i] \phi$ we can easily see that $A_i$ has a 1 in the $(i,j)$ position, so $a_{i,j} = 1$. Thus $F(\phi) \preceq A_i \preceq A$ by monotonicity.

If $a_{k,l} = 1$ then there is some sequence $(i_n, i_{n-1}, \ldots, i_0)$ with $i_n = k$ and $i_0 = l$ such that for each $n \geq j \geq 1$, the matrix $A_j$ has a 1 in the $(i_j, i_{j-1})$ position. By the definition of $A_j$ it is easy to see that either $i_j = i_{j-1}$ or $j \leq i_j \leq i_{j-1} \leq [j] \phi$. If $k = l$ then clearly $f(\phi)_{k,l} = 1$, otherwise $k < l$ so there is at least one $j$ such that $i_j < i_{j-1}$. Let $k'$ be the first such $j$ in the sequence $(i_n, i_{n-1}, \ldots, i_0)$ and let $l'$ be the last such $j$. Then we have:

- $k' \geq l'$,
- $i_{k'} = k$,
- $i_{l'-1} = l$,
- $k' \leq i_{k'} \leq i_{k'-1} \leq [k'] \phi$,
- $l' \leq i_{l'} \leq i_{l'-1} \leq [l'] \phi$.

This gives us $l' \leq k' \leq i_{k'} = k$, then since $\phi$ is order-preserving we get $[l'] \phi \leq [k] \phi$. 
We then have \( k < l = i_{\nu - 1} \leq [l'] \phi \leq [k] \phi \), so from the definition of \( F(\phi) \) we have \( f(\phi)_{k,l} = 1 \). Thus \( A \preceq F(\phi) \), so \( F(\phi) = A \in C^+_n \).

**Example 8.3 (continued)**

With \( \psi \) as before,

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

the matrix \( F(\psi) \) is equal to

\[
F(\psi) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}.
\]

Given \( \phi \in O^+_n \) it is clear from the definition of \( F(\phi) \) that each set \( \{ j : f(\phi)_{i,j} = 1 \} \) is an interval of \( \{1,2,\ldots,n\} \). Since \( \phi \) is order-preserving each set \( \{ i : f(\phi)_{i,j} = 1 \} \) is also an interval of \( \{1,2,\ldots,n\} \), and since \( \phi \) is non-decreasing \( F(\phi) \) is reflexive. A matrix which is reflexive and for which each of the sets \( \{ j : f(\phi)_{i,j} = 1 \} \) and \( \{ i : f(\phi)_{i,j} = 1 \} \) is an interval of \( \{1,2,\ldots,n\} \) is called a **convex** matrix.

**Lemma 8.5**

The set of all convex Boolean matrices is closed under Boolean matrix multiplication.

**Proof.** Given Boolean matrices \( A \) and \( B \), column \( j \) of \( AB \) is equal to the sum of all columns \( i \) of \( A \) such that \( b_{i,j} = 1 \). If \( A \) and \( B \) are convex, then the set \( \{ i : b_{i,j} = 1 \} \) is an interval of \( \{1,2,\ldots,n\} \), so the columns being summed are all adjacent. Each of these columns corresponds to an interval of \( \{1,2,\ldots,n\} \), and since the matrix is reflexive and the columns are adjacent, the union of these intervals is also an
interval of \(\{1, 2, \ldots, n\}\). The sum of these columns corresponds to this union, so corresponds to an interval of \(\{1, 2, \ldots, n\}\). Similarly, each row of \(AB\) corresponds to an interval of \(\{1, 2, \ldots, n\}\). Then, since a product of two reflexive matrices is reflexive, if \(A\) and \(B\) are both convex, \(AB\) is also convex. Therefore the set of convex Boolean matrices is closed under matrix multiplication.

The generators of \(C^+_n\) are convex, so every element of \(C^+_n\) is convex.

**Theorem 8.6**

The map \(F : O^+_n \to C^+_n\) is an isomorphism.

**Proof.** It is clear that \(F\) is injective. It is also not hard to show that it is surjective: we have shown that every element of \(C^+_n\) is convex and upper triangular, and given a convex upper triangular matrix \(A\) the function

\[
[i]_\phi = \max\{j : a_{i,j} = 1\}
\]

clearly maps to \(A\) under \(F\). It remains to show that \(F\) is a homomorphism.

Given \(\phi, \psi \in O^+_n\), let \(A = F(\phi)F(\psi)\). Then we have

\[
a_{i,j} = 1 \iff f(\phi)_{i,k} = f(\psi)_{k,j} = 1 \quad \text{for some } k
\]

\[
\iff i \leq k \leq [i]_\phi \quad \text{and} \quad k \leq j \leq [k]_\psi.
\]

Thus if \(a_{i,j} = 1\) then \(i \leq k \leq j \leq [k]_\psi\), and \(k \leq [i]_\phi\) so since \(\psi\) is order-preserving, \([k]_\psi \leq [i]_\phi [\psi\]. Therefore \(i \leq j \leq [i]_\phi [\psi\) and so \(f_{i,j}(\phi \psi) = 1\).

Conversely, if \(f(\phi \psi)_{i,j} = 1\) then \(i \leq j \leq [i]_\phi [\psi\) and we let \(k = \min\{[i]_\phi, j\}\). Since \(\phi\) is non-decreasing we have \(i \leq [i]_\phi\) and since \(i \leq j\) we have \(i \leq \min\{[i]_\phi, j\} = k\). Since \(\psi\) is non-decreasing we have \(j \leq [j]_\psi\), and since \(j \leq [i]_\phi [\psi\) we have \(j \leq \min\{[i]_\phi [\psi\), \([j]_\psi\}\) which, since \(\psi\) is order-preserving, is equal to \(\min\{[i]_\phi, j\}\) which, since \(\psi\) is order-preserving, is equal to \(\min\{[i]_\phi, j\}\) which, since \(\psi\) is order-preserving, is equal to \(\min\{[i]_\phi, j\}\).

Therefore \(A = F(\phi)F(\psi)\) is equal to \(F(\phi \psi)\) so long as we compose functions from left to right. \(\square\)
Remark 8.7

The monoid of order-preserving reverse parking functions, $O_n^+$, is isomorphic to the upper Catalan monoid, $C_n^+$ by the map $F(\phi) : O_n^+ \to C_n^+$ defined by:

$$f(\phi)_{i,j} = \begin{cases} 
1 & \text{if } i \leq j \leq [i]\phi, \\
0 & \text{otherwise},
\end{cases}$$

with inverse $F^{-1} : C_n^+ \to O_n^+$ defined by:

$$[i]F^{-1}(A) = \max \{ j : a_{i,j} = 1 \}.$$

From now on we shall identify each matrix in $C_n^+$ with the corresponding function $F^{-1}(X)$. We shall treat each element of $C_n^+$ as both a matrix and a function, interchangeably, without further reference to the map $F$.

Similarly, we shall identify each matrix in $C_n^-$ with the corresponding function in $O_n^-$ according to the map $G(\phi) : O_n^- \to C_n^-$ defined by:

$$g(\phi)_{i,j} = \begin{cases} 
1 & \text{if } [i]\phi \leq j \leq i, \\
0 & \text{otherwise},
\end{cases}$$

with inverse $G^{-1} : C_n^- \to O_n^-$ defined by:

$$[i]G^{-1}(A) = \min \{ j : a_{i,j} = 1 \}.$$

In particular, we regard each $\delta_k$ to also be a function which maps $k$ to $k + 1$ and is otherwise equal to the identity function. It is already known that $O_n^+$ is generated by these functions (see [31] for example).

Example 8.3 (continued)

Let $\psi$ be as before,

$$\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\psi & & & & & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}$$
When writing $F(\psi)$ as a product of generators we get $A_6 = \delta_6$, $A_5 = \delta_5$, $A_4 = \delta_4 \delta_5$, $A_2 = \delta_2$ and $A_i = I_8$ for all other $i$. When we regard these as functions their product is

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A_6 = \delta_6 & & & & & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A_5 = \delta_5 & & & & & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A_4 = \delta_4 \delta_5 & & & & & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A_2 = \delta_2 & & & & & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}
\]

which is equal to $\psi$.

In the Catalan monoids, all three decision problems relating to Green’s orders can be solved in polynomial time. We begin with the $L$-order and $R$-order problems.

**Theorem 8.8**

The decision problems CLP and CRP are in $P$.

**Proof.** Given $X, Y \in C_n$, CLP asks if $X \leq_L Y$ and CRP asks if $X \leq_R Y$. Recall that $M^R$ is the result of rotating the matrix $M$ by 180 degrees. Then $X = UY \iff X^{RT} = Y^{RT} U^{RT}$ and, since $C_n$ is the monoid of all upper triangular convex matrices, $A \in C_n \iff A^{RT} \in C_n$. Therefore we have $X \leq_L Y \iff X^{RT} \leq_R Y^{RT}$, so it is enough to prove the result for just CRP.

Given $X, Y \in C_n$ we will define $T \in C_n$, which can be computed in polynomial time, such that $YT = X$ if and only if there exists $V \in C_n$ such that $YV = X$. 

We define $T$ as a function as follows: for each $j \leq n$, if there is some $i$ such that $[i]Y \leq j < [i+1]Y$ then let $[j]T = \max\{j, [i]X\}$. Otherwise let $[j]T = j$. Since $Y$ is order preserving, if such an $i$ exists it is unique, so $T$ is well defined.

We first show that $T \in C_n$ by showing that, as a function, it is non-decreasing and order-preserving. It is non-decreasing since for each $j$ either $[j]T = \max\{j, [i]X\} \geq j$ for some $i$, or $[j]T = j$. To see it is order-preserving, let $j \leq k$ and we will show that $[j]T \leq [k]T$.

Since $T$ is non-decreasing, $k \leq [k]T$. Thus if $[j]T = j$ then $[j]T = j \leq k \leq [k]T$. If $[j]T \neq j$ then $[j]T = [i_j]X$ for some $i_j$ such that $[i_j]Y \leq j < [i_j+1]Y$. If there is no $i_k$ such that $[i_k]Y \leq k < [i_k+1]Y$ then either $k < [1]Y$ or $[n]Y \leq k$, and by the definition of $T$ we have $[k]T = k$. Since $[i_j]Y \leq j \leq k$ and $Y$ is order-preserving, it cannot be the case that $k < [1]Y$, so we have $[n]Y \leq k$. Since $Y$ is non-decreasing, $[n]Y = n$. Thus $[j]T \leq n = [n]Y \leq k = [k]T$.

The remaining possibility is that $[j]T = [i_j]X$ for some $i_j$ such that $[i_j]Y \leq j < [i_j+1]Y$ and there is some $i_k$ such that $[i_k]Y \leq k < [i_k+1]Y$. In this case we must have $i_j \leq i_k$: if we didn’t then we would have $i_k + 1 \leq i_j$ and since $Y$ is order-preserving we would have $k < [i_k+1]Y \leq [i_j]Y \leq j$, a contradiction. Thus $i_j \leq i_k$ and since $X$ is order preserving, $[i_j]X \leq [i_k]X$. Therefore $[j]T = [i_j]X \leq [i_k]X \leq \max\{k, [i_k]X\} = [k]T$. Thus in all cases $[j]T \leq [k]T$.

We now show that $YT = X$ if and only if there exists $V \in C_n$ such that $YV = X$. Clearly if $YT = X$ then there exists such a $V$. For the converse assume there exists $V \in C_n$ such that $YV = X$ and we shall show that $YT = X$.


\[Q.E.D.\]
For the purpose of solving the Catalan $J$-order problem we now describe an algorithm which, given $X,Y \in C_n$, will define $S,T \in C_n$ such that $SYT = X$ if and only if there exist $U,V \in C_n$ such that $UYV = X$. The algorithm will define $S$ and $T$ as functions, and it will always terminate in polynomial time. We will first present the algorithm, then an informal explanation of how it works and an example before proving that it does what we claim.

Algorithm 8.9

1     for $j = 1 : n$ do
2         \[[j]S := j \]
3         \[[j]T := j \]
4     \[C := \{b \le n : [b-1]Y < [b]Y \} \cup \{1\} \]
5     \[b_0 := 0 \]
6     \[k := 0 \]
7     for $j = 1 : n$ do
8         if $j = 1$ or $[j]X > [j-1]X$ then
9             \[k := k + 1 \]
10            \[a_k := j \]
11            \[L := \{b \le n : b \ge \max\{b_{k-1} + 1, a_k\} \text{ and } [b]Y \le [a_k]X \} \]
12            if $L \cap C = \emptyset$ then
13                stop
14            else
15                \[b_k := \min(L \cap C) \]
16     for $i = 1 : k$ do
17         for $j = a_i : b_i$ do
18             \[[j]S := b_i \]
19         for $j = [b_i]Y : [a_i]X$ do
20             \[[j]T := [a_i]X \]
Each operation within this algorithm can clearly be completed in polynomial time. Since each loop iterates for a maximum of $n$ times, the algorithm terminates in polynomial time.

Lines 1 to 3 of the algorithm simply define $S$ and $T$ to both be equal to the identity function. The functions will be redefined later in the algorithm. Initially defining them as the identity function serves two purposes: firstly it allows us to redefine the functions for just the elements of the domain where we want them to differ from the identity function, and secondly it ensures that $S$ and $T$ are defined in cases when the algorithm terminates early. The algorithm only terminates early when there are no $U, V \in C_n$ such that $UYV = X$, and in these cases it doesn’t matter what $S$ and $T$ are defined as, but they must be defined as something.

To understand the purpose of lines 4 to 15 it helps to think of the structure of elements of $C_n$ in matrix form. In an element $A \in C_n$, if there is a 1 at $(i, j)$ which does not have a 1 above or to the right of it then we call $(i, j)$ an upper corner of $A$. Multiplying $A$ on the right by a matrix $B \in C_n$ has the effect of, for each $i \leq n$ simultaneously, copying column $i$ into all columns from $i$ to $[i]B$. This has the effect of moving the upper corners to the right while maintaining convexity (note that some upper corners may cease to be upper corners after this movement). Similarly, multiplying $A$ on the left by $B \in C_n$ has the effect of moving the upper corners upwards while maintaining convexity.

The goal of the algorithm is to define $S$ and $T$ which, in the product $SYT$, will move the upper corners of $Y$ up and right to meet the upper corners of $X$. Lines 4 to 15 of the algorithm find the upper corners of $X$, and assign to each one a suitable upper corner of $Y$ to move to meet that corner of $X$. There are several conditions which must be satisfied for an upper corner of $Y$ to be suitable, and $L$ is the set of values which satisfy these conditions: the condition $b \geq a_k$ ensures that the upper corner of $Y$ is below the upper corner of $X$, the condition
[b]Y \leq [a_k]X ensures that the upper corner of Y is left of the upper corner of X, and the condition \( b \geq b_{k-1} + 1 \) ensures that the upper corner of Y has not already been assigned to another upper corner of X. The set C indexes the rows of Y containing upper corners, so \( L \cap C \) indexes the rows of Y containing suitable upper corners. If \( L \) is empty then there is no \( U, V \in C_n \) such that \( UYV = X \) so the algorithm terminates. Otherwise we choose the first suitable upper corner, to leave as many upper corners as possible available to choose later in the algorithm.

Finally lines 16 to 20 define \( S \) and \( T \) such that multiplying \( Y \) on the left by \( S \) moves each assigned upper corner of \( Y \) up so it is directly left of the upper corner of \( X \) it was assigned to, then multiplying \( SY \) on the right by \( T \) moves the upper corner right to meet the upper corner of \( X \).

We now give an example of the algorithm being applied.

**Example 8.10**

Let \( X, Y \in C_8 \) be the matrices

\[
X = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad Y = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix},
\]

or as functions,

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]
We will run through the algorithm for these matrices. Lines 1 to 3 set $S$ and $T$ to be equal to the identity function. Lines 4 to 6 set

$$C = \{ b \leq 8 : [b-1]Y < [b]Y \} \cup \{1\} = \{1, 3, 5, 7\},$$

$b_0 = 0,$

$k = 0.$

Lines 7 and 8 iterate over all $j$ such that $j = 1$ or $[j]X > [j-1]X$, in ascending order. This condition is equivalent to row $j$ of the matrix $X$ having an upper corner.

The first row of $X$ to have an upper corner is row 1. Lines 9 to 11 set

$$k = 1,$$

$a_1 = 1,$

$$L = \{ b \leq 8 : b \geq \max\{1, 1\} \text{ and } [b]Y \leq [1]X \} = \{1, 2, 3, 4\}.$$  

Now $C \cap L = \{1, 3\}$ because rows 1 and 3 of $Y$ contain upper corners which are below and to the left of the upper corner in row 1 of $X$. Since this is non-empty line 15 sets $b_1 = 1.$
The next row of $X$ to have an upper corner is row 4. Lines 9 to 11 set

\[
k = 2, \\
a_2 = 4, \\
L = \{b \leq 8 : b \geq \max\{2, 4\} \text{ and } [b]Y \leq [4]X\} \\
= \{b \leq 8 : b \geq 4 \text{ and } [b]Y \leq 7\} \\
= \{4, 5, 6\}.
\]

Now $C \cap L = \{5\}$ because row 5 is the only row of $Y$ containing an upper corner which is below and to the left of the upper corner in row 4 of $X$. Since this is non-empty line 15 sets $b_2 = 5$.

The next row of $X$ to have an upper corner is row 5. Lines 9 to 11 set

\[
k = 3, \\
a_2 = 5, \\
L = \{b \leq 8 : b \geq \max\{6, 5\} \text{ and } [b]Y \leq [5]X\} \\
= \{b \leq 8 : b \geq 6 \text{ and } [b]Y \leq 8\} \\
= \{6, 7, 8\}.
\]

Now $C \cap L = \{7\}$ because rows 5 and 7 of $Y$ contain upper corners which are below and to the left of the upper corner in row 5 of $X$, but the upper corner in row 5 of $Y$ is already assigned to another upper corner of $X$. Since this is non-empty line 15 sets $b_3 = 7$.

This defines three pairs, each indexing an upper corner of $X$ and the upper corner of $Y$ assigned to it: $(1, 1), (4, 5)$ and $(5, 7)$. Line 16 now iterates over these pairs.

In the first iteration $a_i = b_i = 1$. Lines 17 and 18 redefine $[1]S = 1$, so $S$ is still equal to the identity map.
Since \([1]Y = 2\) and \([1]X = 4\), lines 19 and 20 redefine \([j]T = 4\) for \(2 \leq j \leq 4\), so now \(T\) is the function

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

In the second iteration \(a_i = 4\) and \(b_i = 5\). Lines 17 and 18 redefine \([j]S = 5\) for \(4 \leq j \leq 5\), so now \(S\) is the function

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Since \([5]Y = 6\) and \([4]X = 7\), lines 19 and 20 redefine \([j]T = 7\) for \(6 \leq j \leq 7\), so now \(T\) is the function

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

In the third iteration \(a_i = 5\) and \(b_i = 7\). Lines 17 and 18 redefine \([j]S = 7\) for \(5 \leq j \leq 7\), so now \(S\) is the function

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Since \([7]Y = 8\) and \([5]X = 8\), lines 19 and 20 redefine \([8]T = 8\), so \(T\) is unchanged.

The algorithm has now finished, and we see that \(SYT = X\) as follows:
or in matrix form,

\[
S = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0
\end{bmatrix},
\]

from which it is easy to verify that \( X = SYT \).

We now prove that the algorithm works as claimed.

**Theorem 8.11**

The decision problem CJP is in P.

**Proof.** Given \( X, Y \in C_n \), CJP asks if \( X \leq_J Y \), or in other words if there exist \( U, V \in C_n \) such that \( UYV = X \).

We will show that if \( S \) and \( T \) are as defined by the above algorithm then \( S, T \in C_n \), and \( SYT = X \) if and only if there exist \( U, V \in C_n \) such that \( UYV = X \).

Since the algorithm terminates in polynomial time this is sufficient to prove the theorem.
We must first show that $S, T \in C_n$, so we need to check that they are both non-decreasing and order-preserving.

To check they are non-decreasing, for each $j \leq n$ we need to check that $[j]S \geq j$ and $[j]T \geq j$. Line 2 defines $[j]S = j$ (so $[j]S \geq j$) and then line 18 redefines it to $[j]S = b_i$ if $a_i \leq j \leq b_i$, so $[j]S \geq j$ in this case too. Line 3 defines $[j]T = j$ (so $[j]T \geq j$) and then line 20 redefines it to $[j]T = [a_i]X$ if $[b_i]Y \leq j \leq [a_i]X$, so $[j]T \geq j$ in this case too. These are the only lines which define or redefine $[j]S$ or $[j]T$. Therefore both functions are non-decreasing.

To show they are order-preserving let $j \leq k$, and we need to check that $[j]S \leq [k]S$ and $[j]T \leq [k]T$. If $[j]S$ is not redefined by line 18 then $[j]S = j \leq k \leq [k]S$. If $[j]S$ has been redefined by line 18 then let $i_j$ be the largest $i$ such that $a_{ij} \leq j \leq b_{ij}$, and then $[j]S = b_{ij}$. If $[k]S$ is not redefined by line 18 then, since $a_{ij} \leq j \leq k$ we must not have $k \leq b_{ij}$. Thus $[j]S = b_{ij} < k \leq [k]S$. If $[k]S$ has been redefined by line 18 then let $i_k$ be the largest $i$ such that $a_{ik} \leq k \leq b_{ik}$, and then $[k]S = b_{ik}$. Since $j \leq k$ we must have $b_{ij} \leq b_{ik}$ (By the condition $b \geq b_{k-1} + 1$ for elements of $L$) and so $[j]S \leq [k]S$. Thus in all cases $[j]S \leq [k]S$ and so $S$ is order-preserving.

The proof that $T$ is order-preserving is analogous.

We now show that $SYT = X$ if and only if there exist $U, V \in C_n$ such that $UYV = X$. Clearly if $SYT = X$ then there exist $U, V \in C_n$ such that $UYV = X$. For the converse assume that $U, V \in C_n$ are such that $UYV = X$ and we will show that $SYT = X$.

We first show that the algorithm doesn’t terminate on line 13 by showing that $L \cap C$ is never empty when the algorithm reaches line 12. For each $i$ we define $m_i = \min\{b \leq n : [b]Y = [a_i]UY\}$. Each time line 12 is reached $k$ has a different value, starting at 1 and increasing by 1 each time. We will prove by induction on $i$ that if $k = i$ when line 12 is reached then $m_i \in L \cap C$. The inductive step will only rely on the inductive hypothesis for $i > 1$ so it also serves at the base case. As our
We inductively assume, for $i > 1$, that $m_{i-1}$ was in $L \cap C$ the last time the algorithm reached line 12.

To show that $m_i \in L \cap C$ we need to show four conditions are satisfied:

1) $m_i \geq b_{i-1} + 1$,

2) $m_i \geq a_i$,

3) $[m_i]Y \leq [a_i]X$,

4) $[m_i - 1]Y < [m_i]Y$ or $m_i = 1$.

If $i = 1$ then $b_{i-1} + 1 = 1$, so condition (1) is satisfied. Otherwise we use the inductive hypothesis to prove condition (1). During the previous iteration of line 12 we had $m_{i-1} \in L \cap C$ and $b_{i-1}$ was defined as $b_{i-1} = \min(L \cap C)$, so $b_{i-1} \leq m_{i-1}$. When $a_{i-1}$ was defined it was defined as $j$, but $j$ was less than its current value, as it was in a previous iteration of the for loop in line 7. Thus $a_{i-1} < j$.

We now have

$$[a_i]X = [j]X$$

since $a_i = j$ by line 10,

$$> [j - 1]X$$

by line 8,

$$\geq [a_{i-1}]X$$

since $a_{i-1} < j$ and $X$ is order-preserving,

$$\Rightarrow [a_i]UYV > [a_{i-1}]UYV$$

since $V$ is order-preserving,

$$\Rightarrow [a_i]UY > [a_{i-1}]UY$$

by the definition of $m_i$,

$$\Rightarrow [m_i]Y > [m_{i-1}]Y$$

since $m_{i-1} \geq b_{i-1}$ and $Y$ is order-preserving,

$$\Rightarrow m_i > b_{i-1}$$

since $Y$ is order-preserving.
Thus condition (1) is satisfied.

We do not require the inductive hypothesis for the other conditions. We have

$$[m_i]U \geq m_i$$ \text{ since } U \text{ is non-decreasing,}

$$\implies [m_i]UY \geq [m_i]Y$$ \text{ since } Y \text{ is order-preserving,}

$$= [a_i]UY$$ \text{ by the definition of } m_i,

$$\implies [m_i]UYV \geq [a_i]UYV$$ \text{ since } V \text{ is order-preserving,}

$$\implies [m_i]X \geq [a_i]X$$ \text{ by lines 10 and 8,}

$$\implies m_i > a_i - 1$$ \text{ since } X \text{ is order-preserving.}

Thus condition (2) is satisfied.

By the definition of $m_i$ we have $[m_i]Y = [a_i]UY$ and since $V$ is non-decreasing we have $[a_i]UY \leq [a_i]UYV = [a_i]X$ so condition (3) is satisfied. If $m_i \neq 1$ then by the definition of $m_i$ we must have $[m_i - 1]Y < [m_i]Y$ so condition (4) is satisfied.

Thus the algorithm does not terminate on line 13. Now we must show that $SYT = X$. Since $a_1 = 1$, given $j \leq n$ we can define $i_j$ as the largest $i$ such that $a_i \leq j$ and we are guaranteed to have $i_j \geq 1$. Since, by lines 8 and 10 of the algorithm, the $a_i$s are the only values for which $[a_i]X > [a_i - 1]X$, we have $[j]X = [a_{i_j}]X$. We split the remainder of the proof into the two cases $j \leq b_{i_j}$ and $j > b_{i_j}$.

First let $j \leq b_{i_j}$. By line 11 we have $[b_{i_j}]Y \leq [a_{i_j}]X$, so lines 19 and 20 define $[b_{i_j}]YT = [a_{i_j}]X$, which we have just seen is equal to $[j]X$. The sequence $(b_1, \ldots, b_k)$ is strictly increasing since each term is defined as $\min(L \cap C)$, and when $b_i$ is defined each element of $L$ is larger than $b_{i-1}$. Since $Y$ is order-preserving the sequence $([b_1]Y, [b_2]Y, \ldots, [b_k]Y)$ is increasing, and by line 4 it is strictly increasing, so $i_j$ is
the largest $i$ such that $[b_i]Y \leq [b_{i_j}]Y \leq [a_i]X$. Lines 19 and 20 therefore do not redefine $[b_{i_j}]YT$ any further after it is defined as $[j]X$.

Since $j \leq b_{i_j}$ we have $a_{i_j} \leq j \leq b_{i_j}$, so lines 17 and 18 define $[j]S = b_{i_j}$. Since $i_j$ is the largest $i$ such that $a_i \leq j$, lines 17 and 18 do not redefine $[j]S$ any further after it is defined as $b_{i_j}$. Therefore $[j]SYT = [b_{i_j}]YT$, which by the previous paragraph is equal to $[j]X$ as required.

Alternatively let $j > b_{i_j}$. Since $Y$ is order-preserving we have $[j]Y \geq [b_{i_j}]Y$. Also

$$j \leq [j]U \quad \text{since $U$ is non-decreasing,}$$
$$\implies [j]Y \leq [j]UY \quad \text{since $Y$ is order-preserving,}$$
$$\leq [j]UYV \quad \text{since $V$ is non-decreasing,}$$
$$= [j]X$$
$$= [a_{i_j}]X.$$

Therefore $[b_{i_j}]Y \leq [j]Y \leq [a_{i_j}]X$ and lines 19 and 20 define $[j]YT = [a_{i_j}]X$, which we have already seen is equal to $[j]X$. Since $i_j$ is the largest $i$ such that $a_i \leq j$, for each $i > i_j$ we have $a_i > j$. Then by line 11 we have $b_i \geq a_i$, so $b_i > j$ and, since $Y$ is order-preserving, $[b_i]Y \geq [j]Y$, and by line 4 this is strict. Therefore $i_j$ is the largest $i$ such that $[b_i]Y \leq [j]Y \leq [a_i]X$ and so lines 19 and 20 do not redefine $[j]YT$ any further after it is defined as $[j]X$.

Since $j > b_{i_j}$ there is no $i$ such that $a_i \leq j \leq b_i$, and $[j]S = j$. Thus $[j]SYT = [j]YT = [j]X$ as required. \qed
Chapter 9

Double Catalan Monoids

The next class of monoids we look at are the double Catalan monoids introduced by Mazorchuk and Steinberg in [26]. The double Catalan monoid of rank $n$ is so named because it can be interpreted as the diagonal submonoid of the direct product of two copies of the corresponding Catalan monoid, regarding one copy as the monoid of non-decreasing order-preserving transformations on $\{1, 2, \ldots, n\}$ and the other as the monoid of non-increasing order-preserving transformations on the same set. We shall formally define this submonoid before defining the double Catalan monoids themselves.

Definition 9.1

We denote by $DO_n$ the submonoid of $C_n^+ \times C_n^-$ generated by $(\delta_k, \delta_k)$ for $1 \leq k \leq n-1$:

$$DO_n = \langle \{ (\delta_k, \delta_k) : 1 \leq k \leq n-1 \} \rangle.$$

Example 9.2

The product $A = (\delta_2, \delta_2)(\delta_3, \delta_3)(\delta_1, \delta_1)(\delta_4, \delta_4)(\delta_2, \delta_2) \in DO_5$ is as follows:
Thus $A$ is equal to

Consider, for each $1 \leq k \leq n - 1$, the $n \times n$ matrix $\varepsilon_k$ defined by

$$(\varepsilon_k)_{i,j} = \begin{cases} 
1 & \text{if } i = j \text{ or } \{i, j\} = \{k, k+1\}, \\
0 & \text{otherwise.} 
\end{cases}$$

These are phone call matrices: $\varepsilon_k = C[k, k+1]$. The monoid generated by these generators is therefore a submonoid of $G_n$. This monoid is the double Catalan monoid of rank $n$:

**Definition 9.3: Double Catalan Monoid of rank $n$**

The double Catalan monoid of rank $n$, denoted $DC_n$, is the monoid generated by
the matrices $\varepsilon_k$ for $1 \leq k \leq n - 1$:

$$DC_n = \langle \{\varepsilon_k : 1 \leq k \leq n - 1\} \rangle.$$  

Since $DC_n \subseteq G_n$, the multiplication simplifies in the same way: right multiplying a matrix $A \in \mathbb{B}_n$ by the matrix $\varepsilon_i$ has the effect of replacing columns $i$ and $i + 1$ of $A$ with the sum of columns $i$ and $i + 1$, while leaving all other columns unchanged, and left multiplying $A$ by $\varepsilon_i$ has the effect of replacing rows $i$ and $i + 1$ of $A$ with the sum of rows $i$ and $i + 1$, while leaving all other rows unchanged.

Each generator of $DC_n$ is convex, so by Lemma 8.5 each element of $DC_n$ is convex.

The main source of results concerning the double Catalan monoids is the paper in which they are defined, [26]. In this paper Mazorchuk and Steinberg give a presentation for $DC_n$ with generating set $\{f_i : 1 \leq i \leq n\}$ and defining relations

$$f_i^2 = f_i \quad \text{for } i \leq n,$$
$$f_if_j = f_jf_i \quad \text{for } |i - j| > 1,$$
$$f_if_{i+1}f_i = f_{i+1}f_if_{i+1} \quad \text{for } i \leq n - 1,$$
$$f_if_{i+1}f_{i+2}f_{i+1}f_i = f_if_{i+1}f_{i+2}f_if_{i+1}f_i \quad \text{for } i \leq n - 2.$$  

The paper also includes some results about a map $D : S_n \rightarrow DC_n$, where $S_n$ is the symmetric group, which is defined on $\{s_1, s_2, \ldots, s_n\}$ by regarding $s_i$ as an element of $\mathbb{B}_n$ in the natural way and defining $D(s_i) = I_n + s_i = \varepsilon_i$, then extended to $S_n$ by, for an arbitrary element $w \in S_n$, taking a reduced decomposition $w = s_{i_1}s_{i_2} \cdots s_{i_k}$ and letting $D(w) = D(s_{i_1})D(s_{i_2}) \cdots D(s_{i_k})$. They show that given $\alpha \in DC_n$, the set $D^{-1}(\alpha)$ contains a unique 4321-avoiding permutation which is the unique Bruhat minimal element in $D^{-1}(\alpha)$, they also show that an element $w \in D^{-1}(\alpha)$ is Bruhat maximal if and only if it is 4321-avoiding and the set $D^{-1}(\alpha)$
is Bruhat convex, and they show that if \( w \in S_n \) is a 4321-avoiding permutation then \( D(w) \) is fixed under transpose if and only if \( w \) is an involution. See [26] for the definitions of 4321-avoiding permutations and the Bruhat order.

It is also shown in [26] that the double Catalan monoid of rank \( n \) is isomorphic to the submonoid of \( C^+_n \times C^-_n \) which we call \( DO_n \). We shall give our own explanation of this isomorphism.

In what follows we will need to multiply elements of \( C^+_n \) by elements of \( C^-_n \). Recall from Remark 8.7 that we regard elements of \( C^+_n \) and \( C^-_n \) as both functions and matrices, interchangeably. We define a product \( \otimes \) as a map from \( C^+_n \times C^-_n \) to \( \bar{G}_n \) as follows: given \( U \in C^+_n \) and \( L \in C^-_n \), regard \( U \) and \( L \) as matrices and let \( U \otimes L \) be the matrix product \( UL \). Since \( C^+_n \) and \( C^-_n \) are both submonoids of \( \bar{G}_n \) the resulting matrix is an element of \( \bar{G}_n \). Since \( DO_n \) is a submonoid of \( C^+_n \times C^-_n \) we can restrict \( \otimes \) to \( DO_n \). We will show that this restriction provides an isomorphism between \( DO_n \) and \( DC_n \).

First we consider the maps \( U : DC_n \rightarrow C^+_n \) and \( L : DC_n \rightarrow C^-_n \) defined by

\[
[k]U(B) = \max\{j : b_{i,j} = 1\},
\]
\[
[k]L(B) = \min\{j : b_{i,j} = 1\}.
\]

Since \( U(\varepsilon_k) = \delta_k \) and \( L(\varepsilon_k) = \hat{\delta}_k \) these maps are both surjective.

**Lemma 9.4**

Given \( B \in DC_n \), when we consider \( U(B) \) and \( L(B) \) as matrices we have \( U(B) \otimes L(B) = U(B) + L(B) = B \).

**Proof.** Let \( i \leq j \). We shall show that position \((i, j)\) is equal to 1 in \( U(B) \otimes L(B) \) if and only if it is equal to 1 in \( U(B) + L(B) \).
Since $U(B) \in C_n^+$ and $L(B) \in C_n^-$, as matrices $U(B)$ is upper unitriangular and $L(B)$ is lower unitriangular. If position $(i, j)$ is equal to 1 in $U(B) + L(B)$ then since $i \leq j$ the same position is equal to 1 in $U(B)$. Therefore, by monotonicity, it is also equal to 1 in $U(B) \otimes L(B)$.

Conversely assume position $(i, j)$ is equal to 1 in $U(B) \otimes L(B)$. Then there is some $k \leq n$ such that $U(B)_{i,k} = L(B)_{k,j} = 1$. Since $L(B)$ is lower triangular we have $k \geq j$. Thus $U(B)_{i,k} = 1$ for some $k$ such that $i \leq j \leq k$, and so by convexity we have $U(B)_{i,j} = 1$, and position $(i, j)$ of $U(B) + L(B)$ is equal to 1.

This shows that $U(B) \otimes L(B)$ is equal to $U(B) + L(B)$ above and on the diagonal. The proof for below the diagonal is entirely analogous.

As a matrix, row $i$ of $U(B)$ has a 1 in exactly the positions $(i, j)$ such that $i \leq j \leq \lfloor i \rfloor U(B) = \max \{ j : b_{i,j} = 1 \}$. This means $U(B)$ is the matrix obtained by setting all below-diagonal entries of $B$ to 0. Similarly, $L(B)$ is the matrix obtained by setting all above-diagonal entries of $B$ to 0. Therefore $B = U(B) + L(B)$.

From the last paragraph of the previous proof we see that $U$ and $L$ are simply the restrictions to $DC_n$ of the maps $U$ and $L$ defined at the start of Chapter 4.

**Theorem 9.5**

The map $\otimes : DO_n \to DC_n$ is an isomorphism.

**Proof.** When we regard $\delta_k$ and $\hat{\delta}_k$ as Boolean matrices and multiply them in $\hat{G}_n$, we get $\delta_k \otimes \hat{\delta}_k = \hat{\delta}_k \otimes \delta_k = \varepsilon_k$. It is easy to check that we still have $\delta_i \otimes \hat{\delta}_j = \hat{\delta}_j \otimes \delta_i$ when $i \neq j$. Therefore, given an element of $DO_n$,

$$A = \prod_{k=1}^{p} (\delta_{i_k} \otimes \hat{\delta}_{i_k}) = \left( \prod_{k=1}^{p} \delta_{i_k} \cdot \prod_{k=1}^{p} \hat{\delta}_{i_k} \right),$$

the image of $A$ under $\otimes$ is

$$\prod_{k=1}^{p} \delta_{i_k} \otimes \prod_{k=1}^{p} \hat{\delta}_{i_k} = \prod_{k=1}^{p} (\delta_{i_k} \otimes \hat{\delta}_{i_k}) = \prod_{k=1}^{p} \varepsilon_{i_k}.$$
This shows that $\otimes$ is a surjective homomorphism. We show it is an isomorphism by giving its inverse. Given $B \in DC_n$ let $\otimes^{-1}(B) = (U(B), L(B))$. This is the inverse of $\otimes$ since, by Lemma 9.4, we have $\otimes \otimes^{-1}(B) = U(B) \otimes L(B) = B$.  

This gives us, for $A, B \in DC_n$,

$$(U(AB), L(AB)) = \otimes^{-1}(AB)$$
$$= \otimes^{-1}(A) \otimes^{-1}(B)$$
$$= (U(A), L(A))(U(B), L(B)) = (U(A)U(B), L(A)L(B)),$$

so we see $U$ and $L$, when restricted to $DC_n$, are homomorphisms.

**Example 9.2 (continued)**

The element $A$ from Example 9.2 is mapped to $\varepsilon_2\varepsilon_3\varepsilon_1\varepsilon_4\varepsilon_2 \in DC_n$ under $\otimes$.

This is equal to the following matrix $B$:

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Since $U(B)$ and $L(B)$ are elements of $C_n^+$ and $C_n^-$ they can be interpreted as functions or as matrices. As functions they are equal to

$$U(B) = \begin{array}{ccccc} & & & & 5 \\ 1 & 2 & 3 & 4 & \end{array} \quad L(B) = \begin{array}{ccccc} & & & & 5 \\ 1 & 2 & 3 & 4 & \end{array}.$$

As matrices they are equal to

$$U(B) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad L(B) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$
These are the matrices obtained by splitting $B$ into upper and lower unitriangular parts.

Elements of the double Catalan monoids are convex, but not every convex $n \times n$ matrix is an element of the double Catalan monoid of rank $n$. Take for example the $2 \times 2$ matrix

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

which is convex but is not equal to $I_2$ or $\varepsilon_1$, which are the only two elements of $DC_2$. It is therefore not always obvious whether or not a given convex matrix is an element of a double Catalan monoid, and so it is worthwhile to consider the membership problem for the double Catalan monoids:

**Definition 9.6: Double Catalan Membership Problem (DCMP)**

Given $C \in B_n$, the double Catalan membership problem asks if $C \in DC_n$.

For the proofs in the rest of this chapter we require a couple of lemmas relating products of generators to the structure of the resulting matrices.

**Lemma 9.7**

Let $A = \prod_{k=1}^{m} \varepsilon_{s_k} \in DC_n$. If $s_k \neq x$ for all $k \leq m$, then $a_{i,j} = 0$ for all $i, j$ satisfying either $i \leq x < j$ or $j \leq x < i$.

**Proof.** Let $i \leq x < j$ and we will show that $a_{i,j} = 0$. The result for $j \leq x < i$ is entirely analogous. Recall that $U(A)$ is the result of replacing all below-diagonal entries in $A$ with 0. Thus $U(A)$ is exactly the same as $A$ above the diagonal, and so $a_{i,j} = 0$ if and only if $U(A)_{i,j} = 0$. If we regard $U(A)$ as a function, then $a_{i,j} = 0$ if and only if $[i]U(A) < j$. 
If \( s_k \neq x \) for all \( k \leq m \) then \( \varepsilon_x = \delta_x \otimes \hat{\delta}_x \) does not occur as a factor of \( A \), and so \( \delta_x \) does not occur as a factor of \( U(A) \). When regarded as functions, \( \delta_x \) is the only generator of \( C^+_n \) which maps an element of \( \{1, \ldots, x\} \) to an element of \( \{x+1, \ldots, n\} \). Thus the function \( U(A) \) cannot map any element of \( \{1, \ldots, x\} \) to any element of \( \{x+1, \ldots, n\} \). Since \( i \leq x \) we have \([i]U(A) \leq x\), and since \( x < j \) we have \([i]U(A) < j\) as required.

Lemma 9.8

Let \( A = \prod_{k=1}^{m} \varepsilon_{s_k} \in DC_n \). If \( i \leq j \), then \( a_{i,j} = 1 \) if and only if \((i, i+1, \ldots, j-1)\) is a scattered subsequence of \((s_1, \ldots, s_m)\). If \( i \geq j \), then \( a_{i,j} = 1 \) if and only if \((i-1, i-2, \ldots, j)\) is a scattered subsequence of \((s_1, \ldots, s_m)\).

Proof. We will prove the lemma only for the case \( i \leq j \). The case \( i \geq j \) is entirely analogous.

It is clear to see that position \((i, j)\) is equal to 1 in \( \varepsilon_i \varepsilon_{i+1} \cdots \varepsilon_{j-1} \), so it is a simple consequence of monotonicity that if \( i, i+1, \ldots, j-1 \) is a scattered subsequence of \( s_1, \ldots, s_m \) then \( a_{i,j} = 1 \). For the converse assume \( a_{i,j} = 1 \). Then \( U(A) = \prod_{k=1}^{m} \delta_{s_k} \in C^+_n \) also has a 1 at \((i, j)\), so as a function we have \( i \leq j \leq [i]U(A) \).

Each generator \( \delta_k \in C^+_n \) maps \( k \) to \( k+1 \) and is otherwise equal to the identity function, so any way of writing \( U(A) \) as a product of generators must include the generators \( \delta_i, \delta_{i+1}, \ldots, \delta_{[i]U(A)-1} \) in that order. Therefore \((i, i+1, \ldots, [i]U(A) - 1)\) is a scattered subsequence of \((s_1, \ldots, s_m)\), and thus so is \((i, i+1, \ldots, j)\).

Next we consider the computational complexity of the decision problems in \( DC_n \). Since \( X \leq_L Y \iff X^T \leq_R Y^T \) and \( DC^T_n = DC_n \) the decision problems DCLP and DCRP have the same computational complexity as each other. We will show that DCMP, DCLP and DCRP are all in P by describing an algorithm which can be used to solve both DCMP and DCRP in polynomial time by factorising elements of \( DC_n \).
Given $X \in DC_n$, the algorithm can produce a sequence of generators of $DC_n$ whose product is $X$, and more generally, given $X \in DC_n$ and a left factor $Y$ of $X$, the algorithm produces a sequence of generators whose product is a matrix $V \in DC_n$ such that $X = YV$.

The algorithm outputs some variables, $r_i^Y$, $l_i^Y$, $r_i^X$, $l_i^X$, $s_i$ and $t_i$, for each $i \leq n$. For each $i \leq n$ we define an ascending sequence of consecutive integers

$$S_i = \begin{cases} (r_i^Y, r_i^Y + 1, \ldots, s_i - 1) & \text{if } s_i > 0, \\ () & \text{if } s_i = 0, \end{cases}$$

and a descending sequence of consecutive integers

$$T_i = \begin{cases} (t_i - 1, t_i - 2, \ldots, l_i^X) & \text{if } t_i > 0, \\ () & \text{if } t_i = 0. \end{cases}$$

We concatenate these sequences to form $V = S_n S_{n-1} \cdots S_1 T_1 T_2 \cdots T_n$, and then $V$ is the product of generators corresponding to the sequence $V$. For an arbitrary sequence of integers $A$ we shall use the notation

$$\varepsilon_A = \prod_{j \in A} \varepsilon_j.$$ 

With this notation, $V = \varepsilon_V$.

The algorithm will always terminate in polynomial time and output the relevant variables, even if given incorrect input (Boolean matrices which are not in $DC_n$, or $X, Y \in DC_n$ such that $Y$ is not a left factor of $X$) although the matrix $V$ is then not guaranteed to satisfy any particular equation. This is important as it will allow us to use the algorithm to solve the decision problems DCMP and DCRP in polynomial time.

We will first present the algorithm, then give an informal description of how it works and a couple of examples before proving that $X = YV$ as claimed.
Algorithm 9.9

1 for $i = 1 : n$ do
2 \hspace{1em} $l^X_i := \min\{j : x_{i,j} = 1\}$
3 \hspace{1em} $r^X_i := \max\{j : x_{i,j} = 1\}$
4 \hspace{1em} $l^Y_i := \min\{j : y_{i,j} = 1\}$
5 \hspace{1em} $r^Y_i := \max\{j : y_{i,j} = 1\}$
6 for $i = 1 : n$ do
7 \hspace{2em} $s_i := 0$
8 \hspace{2em} $t_i := 0$
9 \hspace{2em} if ($r^X_i > r^X_{i-1}$ or $i = 1$) and $r^X_i > r^Y_i$ then
10 \hspace{3em} $s_i := r^X_i$
11 \hspace{2em} if ($l^X_i < l^X_{i+1}$ or $i = n$) and $l^X_i < l^Y_i$ then
12 \hspace{3em} $t_i := l^Y_i$
13 \hspace{2em} $p := \max\{i : s_i > 0 \text{ or } i = 0\}$
14 \hspace{2em} $q := \max\{j : t_j > 0 \text{ or } j = 0\}$
15 \hspace{2em} while $p > 0$ and $q > 0$ do
16 \hspace{3em} if $s_p \geq t_q$ then
17 \hspace{4em} for $j = 1 : q$ do
18 \hspace{5em} if $r^Y_p \leq t_j < s_p$ then
19 \hspace{6em} $t_j := t_j - 1$
20 \hspace{5em} if $t_j = l^X_j$ then
21 \hspace{6em} $t_j := 0$
22 \hspace{4em} $p := \max\{i < p : s_i > 0 \text{ or } i = 0\}$
23 \hspace{4em} $q := \max\{j \leq q : t_j > 0 \text{ or } j = 0\}$
24 \hspace{3em} else
25 \hspace{4em} for $i = 1 : p$ do
26 \hspace{5em} if $l^X_q \leq s_i < t_q$ then
27 \hspace{6em} $s_i := s_i - 1$
28 \hspace{5em} if $s_i = r^Y_i$ then
29 \hspace{6em} $s_i := 0$
30 \hspace{5em} $p := \max\{i \leq p : s_i > 0 \text{ or } i = 0\}$
31 \hspace{5em} $q := \max\{j < q : t_j > 0 \text{ or } j = 0\}$
Each operation within this algorithm can clearly be completed in polynomial time. Each for loop iterates for a maximum of \( n \) times, so to see that the algorithm terminates in polynomial time it is enough to check that the number of iterations of the while loop in line 15 is bounded above by a polynomial in \( n \). The highest possible values of \( p \) and \( q \) is \( n \), and on each iteration of the while loop one of these values is lowered and the other isn’t increased. The while loop terminates once one of these values reaches 0, which will take no more than \( 2n \) iterations.

The algorithm is split into two parts: the first part, lines 1 to 12, defines the variables \( t_i^X, r_i^X, t_i^Y \) and \( r_i^Y \) which describe the structure of \( X \) and \( Y \) by locating the left-most and right-most 1 in each row. Since \( X \) and \( Y \) are convex these variables are enough to fully describe them. It then defines the variables \( s_i \) and \( t_i \) which, together with \( r_i^Y \) and \( t_i^X \), encode the matrix \( V \). At this point \( V \) is just the algorithm’s “first attempt” at constructing a suitable \( V \) and is unlikely to satisfy \( X = YV \). The second part of the algorithm, lines 13 to 31, then makes several passes through a while loop which decreases the variables \( s_i \) and \( t_i \), and creates several more attempts at a suitable \( V \), the last of which will satisfy \( X = YV \).

Each pass through the while loop on line 15 lowers each variable \( s_i \) and \( t_j \) by at most 1 but does not change the variables \( r_i^Y \) and \( l_i^X \). Lowering \( s_i \) or \( t_j \) by 1 is equivalent to removing the largest term from the corresponding sequence \( S_i \) or \( T_j \). If all terms are removed from a sequence then the algorithm sets the corresponding variable to 0 (lines 20-21 and 28-29) to indicate that the sequence is empty.

In an element \( A \in DC_n \), if there is a 1 at \((i, j)\) which does not have a 1 above or to the right of it then we call \((i, j)\) an upper corner of \( A \). If there is a 1 at \((i, j)\) which does not have a 1 below or to the left of it then we call \((i, j)\) a lower corner of \( A \). Since elements of \( DC_n \) are convex, all upper corners are above the diagonal and all lower corners are below the diagonal.
Right multiplication of \( A \) by a generator \( \varepsilon_i \) can be seen as moving any upper corner in column \( i \) of \( A \) right one column, and any lower corner in column \( i + 1 \) of \( A \) left one column. Since right multiplication can only move the corners left or right, and not up or down, the fact that \( Y \) is a left factor of \( X \) means that for each upper corner of \( X \) there is an upper corner of \( Y \) in the same row.

The first part of the algorithm (lines 1 to 12) identifies which corners of \( Y \) need to be moved left or right in order to result in \( X \), and for each such corner finds a matrix which can move it to the correct position by right multiplication.

The conditional in line 9 checks if \( (r_i^X > r_{i-1}^X \text{ or } i = 1) \) and \( r_i^X > r_i^Y \). The first part, \( r_i^X > r_{i-1}^X \text{ or } i = 1 \), is true if and only if \( (i, r_i^X) \) is an upper corner of \( X \). For each upper corner of \( X \) the second part of the conditional, \( r_i^X > r_i^Y \), checks whether the corresponding upper corner of \( Y \) is already in the correct place, or whether it needs to be moved into place. If it does need to be moved into place, then line 10 sets the sequence \( S_i = (r_i^Y, r_i^Y + 1, \ldots, r_i^X - 1) \) by setting \( s_i = r_i^X \). We will soon show that right multiplying \( Y \) by \( \varepsilon S_i \) does the job of moving the corner of \( Y \) to the location of the corner of \( X \).

Similarly, the conditional in line 11 checks for lower corners of \( X \) such that the corresponding lower corner of \( Y \) needs to be moved into place, and line 12 sets \( T_i = (l_i^Y - 1, l_i^Y - 2, \ldots, l_i^X) \) by setting \( t_i = l_i^Y \). Then right multiplying \( Y \) by \( \varepsilon T_i \) performs the job of moving the corner of \( Y \) to the location of the corner of \( X \).

Individually, the sequences defined by the first part of the algorithm move each corner to the correct location. However, when the sequences are concatenated and \( Y \) is right multiplied by the corresponding product of generators, \( V \), the sequences may interfere with each other and move the corners too far. The second part of the algorithm fixes this by shortening the sequences so that right multiplying \( Y \) by \( V \) moves each corner to the desired position and no further.
Let \( Y \) be any sequence such that \( \varepsilon_Y = Y \). For any sequence \( A \) we have \( Y \varepsilon_A = \varepsilon_Y A \). Thus by Lemma 9.8 there is a 1 at position \((i, j)\) of \( Y \varepsilon_A \) if and only if \((i, i + 1, \ldots, j - 1)\) is a scattered subsequence of \( Y A \) for \( i \leq j \), or \((i - 1, i - 2, \ldots, j)\) is a scattered subsequence of \( Y A \) for \( i \geq j \).

Let \((i, r_Y^i)\) be an upper corner of \( Y \) which needs to be moved to \((i, r_X^i)\). By Lemma 9.8 we know that \( Y \) must have \((i, i + 1, \ldots, r_Y^i - 1)\) as a scattered subsequence, but not \((i, i + 1, \ldots, r_Y^i)\). The algorithm initially defines \( S_i = (r_Y^i, r_Y^i + 1, \ldots, r_X^i - 1) \), so at this point \( Y S_i \) has \((i, i + 1, \ldots, r_X^i - 1)\) as a scattered subsequence but not \((i, i + 1, \ldots, r_X^i)\), which shows that right multiplying \( Y \) by \( \varepsilon S_i \) does indeed move the corner to the correct position.

We can see that \((i, i + 1, \ldots, r_X^i)\) is not a scattered subsequence of \( Y S_n S_{n-1} \cdots S_1 \) as follows: for each \( j > i \) such that \( S_j \) is non-empty we have \( r_Y^j > r_X^i \) (since \( S_j \) is non-zero there is an upper corner in row \( j \) of \( X \), and so there is an upper corner in row \( j \) of \( Y \), then the inequality follows from convexity) so \( S_j \), whose lowest term is \( r_Y^j \), doesn’t contain \( r_X^i \), and therefore \((i, i + 1, \ldots, r_Y^i, \ldots, r_X^i)\) isn’t a scattered subsequence of \( Y S_n S_{n-1} \cdots S_{i+1} \). For each \( j \leq i \) we have \( r_X^j \leq r_X^i \) (again by convexity) so \( S_j \), whose largest term is \( r_X^j - 1 \), doesn’t contain \( r_X^i \), and therefore \((i, i + 1, \ldots, r_X^i)\) is not a scattered subsequence of \( Y S_n S_{n-1} \cdots S_1 \).

Thus \((i, i + 1, \ldots, r_X^i - 1)\) is a scattered subsequence of \( Y S_n S_{n-1} \cdots S_1 \) but \((i, i + 1, \ldots, r_X^i)\) is not. Since this is true for all upper corners which need to be moved into place it follows that \( \varepsilon Y S_n S_{n-1} \cdots S_1 \) is the same as \( X \) above the diagonal. Similar considerations show that \( \varepsilon Y T_1 T_2 \cdots T_n \) is the same as \( X \) below the diagonal.

However, for a given ascending sequence \( S_i = (r_Y^i, r_Y^i + 1, \ldots, r_X^i - 1) \) it is possible that a descending sequence \( T_j \) will contain \( r_X^i \), and then \( \varepsilon_Y V \) will contain \((i, i + 1, \ldots, r_X^i)\) as a scattered subsequence, so the upper corner in row \( i \) of \( Y \) will be moved too far by right multiplication by \( V \). We will show in Lemma 9.12 that if \( r_X^i \in T_j \) then \( r_X^i - 1 \in T_j \). Therefore we can remove the term \( r_X^i - 1 \)
from $\mathcal{S}_i$ with the intention that $\mathcal{T}_j$ will now contribute the term $r_i^X - 1$ to the scattered subsequence $(i, i+1, \ldots, r_i^X - 1)$ of $V$ instead of contributing an unwanted extra term. However it is possible that a descending sequence $\mathcal{T}_{j'}$ with $j' < j$ contains $r_i^X - 1$, in which case $\varepsilon_{YV}$ will still contain $(i, i+1, \ldots, r_i^X)$ as a scattered subsequence. Again, we will show that if $r_i^X - 1 \in \mathcal{T}_{j'}$ then $r_i^X - 2 \in \mathcal{T}_{j'}$, so we can remove the term $r_i^X - 2$ from $\mathcal{S}_i$ with the intention that $\mathcal{T}_{j'}$ will contribute the term $r_i^X - 2$ to the scattered subsequence $(i, i+1, \ldots, r_i^X - 1)$, and then $\mathcal{T}_j$ will contribute the term $r_i^X - 1$ instead of adding an unwanted extra term. In this fashion we can continue to remove terms from $\mathcal{S}_i$ until this intention is achieved.

The second part of the algorithm checks for conflicts such as this and removes the largest term of $\mathcal{S}_i$ in each case. Similarly, it also checks for the situation where an $\mathcal{S}_i$ sequence adds an unwanted extra term to a $\mathcal{T}_j$ sequence, and removes that term from $\mathcal{T}_j$. What is done in each iteration of the while loop in line 15 depends on the values of $p$ and $q$ and whether or not $s_p \geq t_q$. If $s_p \geq t_q$ then the largest term is removed from each $\mathcal{T}_j$ if and only if it is one less than a term of $\mathcal{S}_p$. We call this the $\mathcal{S}_p$ iteration. If $s_p < t_q$ then the largest term is removed from each $\mathcal{S}_i$ if and only if it is one less than a term of $\mathcal{T}_q$. We call this the $\mathcal{T}_q$ iteration.

We give two examples of the algorithm being used. We first give an example where $Y = I_n$, so the algorithm tells us how to write $X$ as a product of generators.

**Example 9.10**

Let $X \in DC_5$ be

\[
X = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 \\
o & 1 & 1 & 1 & 1 \\
o & 1 & 1 & 1 & 1 \\
o & 0 & 1 & 1 & 1
\end{bmatrix},
\]

and let $Y = I_5$. 
The first 12 lines of the algorithm define $r^X_i, l^X_i, r^Y_i, l^Y_i, s_i$ and $t_i$ for $1 \leq i \leq 5$ as follows:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r^X_i$</th>
<th>$l^X_i$</th>
<th>$r^Y_i$</th>
<th>$l^Y_i$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

The variables $s_1$ and $s_3$ are defined as such because rows 1 and 3 of $X$ contain upper corners. The variables $t_2, t_4$ and $t_5$ are defined as such because rows 2, 4 and 5 of $X$ contain lower corners. Lines 13 and 14 now set $p = 3$ and $q = 5$. This is indicated by the positions of the circles in the table above.

In the first iteration of the while loop we have $s_p = t_q$, so this is the $S_3$ iteration and the algorithm runs through lines 17 to 23. Lines 17 and 18 check, for each $j \leq 5$, whether or not $3 \leq t_j < 5$. At this point we have $S_3 = (3, 4)$, so the algorithm is checking, for each $j \leq 5$, whether or not $t_j \in S_3$. If $t_j \in S_3$ then $t_j - 1$ must be removed from $T_j$ to avoid interference when the sequences are concatenated. This is done by lowering $t_j$ by 1. We have $t_4 = 4 \in S_3$, so line 19 sets $t_4 = 3$. Now lines 22 and 23 set $p = 1$ and $q = 5$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r^X_i$</th>
<th>$l^X_i$</th>
<th>$r^Y_i$</th>
<th>$l^Y_i$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

In the second iteration we have $s_p < t_q$, so this is the $T_5$ iteration and the
algorithm runs through lines 25 to 31. Lines 25 and 26 check, for each $i \leq 1$, whether or not $3 \leq s_i < 5$. At this point we have $T_5 = (4, 3)$, so the algorithm is checking, for each $i \leq 1$, whether or not $s_i \in T_5$. This is true for $s_1 = 4$, so line 27 sets $s_1 = 3$. Now lines 30 and 31 set $p = 1$ and $q = 4$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^X$</th>
<th>$l_i^X$</th>
<th>$r_i^Y$</th>
<th>$l_i^Y$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

In the third iteration we have $s_p = t_q$, so this is the $S_1$ iteration and the algorithm runs through lines 17 to 23. Lines 17 and 18 check, for each $j \leq 4$, whether $1 \leq t_j < 3$. At this point we have $S_1 = (1, 2)$, so the algorithm checks, for each $j \leq 4$, whether $t_j \in S_1$. This is true for $t_2 = 2$, so line 19 sets $t_2 = 1$, and then $t_2 = l_2^X$ so line 21 sets $t_2 = 0$. Now lines 22 and 23 set $p = 0$ and $q = 4$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^X$</th>
<th>$l_i^X$</th>
<th>$r_i^Y$</th>
<th>$l_i^Y$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Now the algorithm terminates as $p = 0$. The final values of $s_i$ and $t_i$ give us

$$S_1 = (1, 2), \quad S_3 = (3, 4),$$

$$T_4 = (2), \quad T_5 = (4, 3),$$
and all other sequences are empty. Thus $V = S_5S_4S_3S_2S_1T_1T_2T_3T_4T_5$ is equal to $(3, 4, 1, 2, 2, 4, 3)$. The algorithm now tells us that $X = \varepsilon_V = \varepsilon_3\varepsilon_4\varepsilon_1\varepsilon_2\varepsilon_2\varepsilon_4\varepsilon_3$, and it is easy to verify that this is correct.

Next we give an example where $Y \neq I_n$ and the algorithm defines a $V \in DC_n$ such that $X = YV$.

**Example 9.11**

Let $X, Y \in DC_{10}$ be

$$X = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{bmatrix}, \quad Y = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

The first 12 lines of the algorithm define $r_i^X, l_i^X, r_i^Y, l_i^Y, s_i$ and $t_i$ for $1 \leq i \leq 10$ as follows:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^X$</th>
<th>$l_i^X$</th>
<th>$r_i^Y$</th>
<th>$l_i^Y$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The variables $s_1, s_6$ and $s_7$ are defined as such because rows 1, 2 and 7 of $X$ contain upper corners. The variables $t_4, t_5$ and $t_{10}$ are defined as such because rows 4, 5 and 10 of $X$ contain lower corners. The algorithm sets $t_8 = 0$ even...
though row 8 of \( X \) has a lower corner because \( l_{8}^{X} = l_{8}^{Y} \). Lines 13 and 14 now set \( p = 7 \) and \( q = 10 \).

In the first iteration of the while loop we have \( s_{p} = t_{q} \), so this is the \( S_{7} \) iteration. At this point we have \( S_{7} = (9) \) and the algorithm checks, for each \( j \leq 10 \), whether or not \( t_{j} \in S_{7} \). This is not the case for any of the \( t_{j} \) so none of them are changed. Lines 22 and 23 now set \( p = 6 \) and \( q = 10 \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( r_{i}^{X} )</th>
<th>( l_{i}^{X} )</th>
<th>( r_{i}^{Y} )</th>
<th>( l_{i}^{Y} )</th>
<th>( s_{i} )</th>
<th>( t_{i} )</th>
<th>( i )</th>
<th>( r_{i}^{X} )</th>
<th>( l_{i}^{X} )</th>
<th>( r_{i}^{Y} )</th>
<th>( l_{i}^{Y} )</th>
<th>( s_{i} )</th>
<th>( t_{i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

In the second iteration we have \( s_{p} < t_{q} \), so this is the \( T_{10} \) iteration. At this point we have \( T_{10} = (9, 8, 7) \) and the algorithm checks, for each \( i \leq 7 \), whether or not \( s_{i} \in T_{10} \). This is true for \( s_{6} = 9 \), so line 27 sets \( s_{6} = 8 \), and then \( s_{6} = r_{6}^{Y} \) so line 29 sets \( s_{6} = 0 \). Now lines 30 and 31 set \( p = 1 \) and \( q = 5 \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( r_{i}^{X} )</th>
<th>( l_{i}^{X} )</th>
<th>( r_{i}^{Y} )</th>
<th>( l_{i}^{Y} )</th>
<th>( s_{i} )</th>
<th>( t_{i} )</th>
<th>( i )</th>
<th>( r_{i}^{X} )</th>
<th>( l_{i}^{X} )</th>
<th>( r_{i}^{Y} )</th>
<th>( l_{i}^{Y} )</th>
<th>( s_{i} )</th>
<th>( t_{i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

In the third iteration, \( s_{p} > t_{q} \), so this is the \( S_{1} \) iteration. At this point we have \( S_{1} = (2, 3, 4) \) and the algorithm checks, for each \( j \leq 5 \), whether or not
$t_j \in \mathcal{S}_1$. This is true for $t_4 = 3$ and $t_5 = 4$, so line 19 sets $t_4 = 2$ and $t_5 = 3$.

Now lines 22 and 23 set $p = 0$ and $q = 5$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^X$</th>
<th>$l_i^X$</th>
<th>$r_i^Y$</th>
<th>$l_i^Y$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^X$</th>
<th>$l_i^X$</th>
<th>$r_i^Y$</th>
<th>$l_i^Y$</th>
<th>$s_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Now the algorithm terminates since $p = 0$. These final values of $s_i$ and $t_i$ give us

$\mathcal{S}_1 = (2, 3, 4), \quad \mathcal{S}_7 = (9), \quad \mathcal{T}_4 = (1), \quad \mathcal{T}_5 = (2), \quad \mathcal{T}_{10} = (9, 8, 7)$,

and all other sequences are empty. Thus $V = \mathcal{S}_{10}\mathcal{S}_9 \cdots \mathcal{S}_1 \mathcal{T}_1 \mathcal{T}_2 \cdots \mathcal{T}_{10}$ is equal to $(9, 2, 3, 4, 1, 2, 9, 8, 7)$. Therefore $Y \varepsilon_Y = Y\varepsilon_9\varepsilon_2\varepsilon_3\varepsilon_4\varepsilon_1\varepsilon_2\varepsilon_9\varepsilon_8\varepsilon_7$, and it is easy to verify that this is equal to $X$.

We will now prove that $X = YV$, and we will use the $X$ and $Y$ from the previous example as a running example. We start with a lemma which will be used in the proof itself. For this lemma and proof, when we talk about the variables in the algorithm or the sequences represented by these variables we refer to their values at the end of the algorithm unless explicitly stated otherwise.

**Lemma 9.12**

Given $X, Y \in DC_n$, with the variables defined by the above algorithm and with each $\mathcal{S}_i$ and $\mathcal{T}_i$ defined as above, if $Y$ is a left factor of $X$, then the following statements hold:
1) An element of $DC_n$ cannot have an upper corner and a lower corner in the same column unless that column is equal to a unit vector. Thus, if $r_i^X = l_j^X$ then $r_i^X = l_j^X = i = j$ and if $r_i^Y = l_j^Y$ then $r_i^Y = l_j^Y = i = j$.

2) For each row of $X$ which contains an upper or lower corner, the same row in $Y$ contains the same type of corner. In other words, for any $i,j \leq n$, if $r_i^X > r_j^X$ then $r_i^Y > r_j^Y$ and if $l_i^X < l_j^X$ then $l_i^Y < l_j^Y$.

3) The sequences $(r_1^X, r_2^X, \ldots, r_n^X)$ and $(r_1^Y, r_2^Y, \ldots, r_n^Y)$ are non-decreasing. The subsequences of $(r_1^X, r_2^X, \ldots, r_n^X)$ and $(r_1^Y, r_2^Y, \ldots, r_n^Y)$ consisting of just the terms which correspond to the non-empty $S_i$ are strictly increasing.

The sequences $(l_1^X, l_2^X, \ldots, l_n^X)$ and $(l_1^Y, l_2^Y, \ldots, l_n^Y)$ are also non-decreasing. The subsequences of $(l_1^X, l_2^X, \ldots, l_n^X)$ and $(l_1^Y, l_2^Y, \ldots, l_n^Y)$ consisting of just the terms which correspond to the non-empty $T_j$ are also strictly increasing.

At any point in the algorithm, the non-zero terms of the sequences $(s_1, s_2, \ldots, s_n)$ and $(t_1, t_2, \ldots, t_n)$ form non-decreasing sequences.

4) No terms are removed from $S_p$ after the $S_p$ iteration, and no terms are removed from $T_q$ after the $T_q$ iteration.

5) For any $q$ and $i$, if $s_i$ is in $T_q$ immediately before the $T_q$ iteration then so is $s_i - 1$. In other words, since $l_i^X$ is the smallest term in $T_q$, immediately before the $T_q$ iteration we have $s_i \neq l_i^X$ for all $i$.

For any $p$ and $j$, if $t_j$ is in $S_p$ immediately before the $S_p$ iteration then so is $t_j - 1$. In other words, immediately before the $S_p$ iteration we have $t_j \neq r_j^Y$ for all $j$.

6) If, at any point during the algorithm, $B = (b_1, b_2, \ldots, b_m)$ is a scattered subsequence of $S_n S_{n-1} \cdots S_1$ consisting of consecutive decreasing integers, and column $b_1 + 1$ of $Y$ contains a lower corner, then $B$ is a subsequence of
some $T_j$ when it is initially defined.

If, at any point during the algorithm, $B = (b_1, b_2, \ldots, b_m)$ is a scattered subsequence of $T_1 T_2 \cdots T_n$ consisting of consecutive increasing integers, and column $b_1$ of $Y$ contains an upper corner, then $B$ is a subsequence of some $S_i$ when it is initially defined.

Proof. 1) Let $A \in DC_n$ and assume that $(i_1, j)$ is an upper corner of $A$ and $(i_2, j)$ is a lower corner of $A$. Column $j$ has a 1 in row $i_1$ but by convexity no columns to the right of it have a 1 in this row, so column $j$ is not $\preceq$-below or equal to any of the columns to its right. Column $j$ also has a 1 in row $i_2$ but no columns to the left of it have a 1 in this row, so it is also not $\preceq$-below or equal to any column to its left. Thus column $j$ is unique and $\preceq$-maximal among the columns of $A$, which means, since $A \in G_n$ and by Lemma 6.8, that column $j$ is the $j$th unit vector.

If $r_i^X = l_j^X$ for some $i$ and $j$ then let $c = r_i^X = l_j^X$. Then there is an upper corner and a lower corner in column $c$ of $X$ so it is equal to the $c$th unit vector. Then since $(i, c) = (i, r_i^X) = 1$ we must have $i = c$, and since $(j, c) = (j, l_j^X) = 1$ we must have $j = c$. Therefore $r_i^X = l_j^X = c = i = j$.

Similarly if $r_i^Y = l_j^Y$ for some $i$ and $j$ then $r_i^Y = l_j^Y = i = j$.

2) We know there exists $V \in DC_n$ such that $X = YV$. Let $\mathcal{Y}$ and $\mathcal{V}$ be sequences such that $Y = \varepsilon_\mathcal{Y}$ and $V = \varepsilon_\mathcal{V}$. Assume that $Y$ doesn’t have an upper corner in row $i$ and we will show that $X$ also doesn’t have an upper corner in row $i$. Since $Y$ doesn’t have an upper corner in row $i$ we have $y_i, r_i^X = 1$. Since $x_i, r_i^Y = 1$ we know, by Lemma 9.8, that $(i, i + 1, \ldots, r_i^X - 1)$ is a scattered subsequence of $\mathcal{Y} \mathcal{V}$. Since $r_i^Y = \max \{ j : y_i, j = 1 \}$, either $y_i, r_i^Y + 1 = 0$ or $r_i^Y = n$. By Lemma 9.8 and the fact that $n - 1$ is the largest number indexing a generator of $DC_n$, either way $(i, i + 1, \ldots, r_i^Y)$ isn’t a scattered subsequence.
of \( \mathcal{Y} \). Thus \((r_i^Y, r_i^Y + 1, \ldots, r_i^X - 1)\) must be a scattered subsequence of \( \mathcal{Y} \). Then, since \( y_{i-1,r_i^Y} = 1 \) we know that \((i - 1, i, \ldots, r_i^Y - 1)\) is a scattered subsequence of \( \mathcal{Y} \) and so \((i - 1, i, \ldots, r_i^X - 1)\) is a scattered subsequence of \( \mathcal{Y} \mathcal{V} \). Then Lemma 9.8 gives \( x_{i-1,r_i^X} = 1 \), so row \( i \) of \( X \) does not have an upper corner.

Thus if \( X \) has an upper corner in row \( i \) then so does \( Y \). If \( r_i^X > r_j^X \) then there must be an upper corner somewhere between rows \( i \) and \( j \) in \( X \), and so there must be an upper corner somewhere between rows \( i \) and \( j \) in \( Y \), and thus \( r_i^Y > r_j^Y \). The proof for lower corners is similar.

3) It immediately follows from convexity that the sequences \((r_1^X, r_2^X, \ldots, r_n^X)\) and \((r_1^Y, r_2^Y, \ldots, r_n^Y)\) are non-decreasing. Line 9 of the algorithm contains an if statement which, for \( i > 1 \), is only true if \( r_i^X > r_{i-1}^X \), and \( S_i \) is only set to be non-empty if this if statement is true. Thus the terms of the sequence \((r_1^X, r_2^X, \ldots, r_n^X)\) corresponding to the non-empty \( S_i \) form a strictly increasing sequence. It immediately follows from Lemma 9.12 part 2 that the terms of \((r_1^Y, r_2^Y, \ldots, r_n^Y)\) corresponding to the non-empty \( S_i \) also form a strictly increasing sequence.

The proof for the \((l_1^X, l_2^X, \ldots, l_n^X)\) and \((l_1^Y, l_2^Y, \ldots, l_n^Y)\) sequences is entirely analogous.

For the last part, we note that each \( s_i \) is initially defined as either 0 or \( r_i^X \), so the non-zero terms form a non-decreasing sequence since \((r_1^X, r_2^X, \ldots, r_n^X)\) is non-decreasing. Assume for contradiction that at some point later in the algorithm we have \( 0 < s_j < s_i \) for some \( i < j \). Since we initially had \( s_i \leq s_j \), and the algorithm cannot increase the value of \( s_i \), it must have decreased the value of \( s_j \) to arrive at this state. The only time that \( s_j \) is decreased is on a \( T_q \) iteration, where it is decreased by 1 if and only if \( s_j \in T_q \). There must
therefore be some $q$ such that $s_i = s_j$ before the $T_q$ iteration, but then the $T_q$ iteration lowers $s_j$ but doesn’t lower $s_i$. However, this means that immediately before the $T_q$ iteration we have $s_j \in T_q$ and $s_i \notin T_q$, contradicting $s_i = s_j$. This shows that the non-zero terms of $(s_1, s_2, \ldots, s_n)$ are non-decreasing, and the proof for $(t_1, t_2, \ldots, t_n)$ is entirely analogous.

4) The only lines of the algorithm which can change the values of $p$ and $q$ are lines 13, 14, 22, 23, 30, and 31. Looking at these lines, we see that $p$ and $q$ are always either set to values such that $s_p$ and $t_q$ are non-zero, or set to zero in which case the algorithm ends immediately after $p$ and $q$ have been updated. The only parts of the algorithm where a non-zero $s_p$ or $t_q$ can be decreased to zero are lines 21 and 29, and in both of these cases the values of $p$ and $q$ are updated immediately afterwards. Thus, at the start of each iteration of the while loop in line 15, $s_p$ and $t_q$ are non-zero.

Lines 13, 14, 22, 23, 30, and 31 also show that $p$ and $q$ never increase after their initial definition. Thus, since the terms $s_p$ and $t_q$ are non-zero at the start of each iteration, and the non-zero terms of $(s_1, s_2, \ldots, s_n)$ and $(t_1, t_2, \ldots, t_n)$ form non-decreasing sequences (by Lemma 9.12 Part 3), at the start of each iteration the values of $s_p$ and $t_q$ are no larger than they were at the start of the previous iteration.

On each iteration of the while loop, the algorithm checks if $s_p \geq t_q$ (in which case the largest term of $S_p$ is at least as large as the largest term of $T_q$) and if so, the iteration is the $S_p$ iteration. If not, then $s_p < t_q$ (so the largest term of $T_q$ is larger than the largest term of $S_p$), and the iteration is the $T_q$ iteration. The while loop is therefore iterating over the non-empty $S_i$ and $T_j$ sequences in order of decreasing largest term.

Therefore, for a given $p$, for each $T_q$ iteration occurring after the $S_p$ iteration,
CHAPTER 9. DOUBLE CATALAN MONOIDS

$s_p$, which is greater than the largest term in $S_p$, cannot be a term of $T_q$. Thus no terms are removed from $S_p$ after the $S_p$ iteration. Entirely analogous arguments show that no terms are removed from $T_q$ after the $T_q$ iteration.

5) If $S_i$ is non-empty then the largest term of $S_i$ is $s_i - 1$, and $s_i$ is initially defined as $r^X_i$. Thus, immediately before the $T_q$ iteration either $s_i = l^Y_i$ or $S_i$ has previously had the terms $(s_i, s_i + 1, \ldots, r^X_i - 1)$ removed from it. If $s_i = r^X_i$ then we cannot have $s_i = l^Y_i$: if we did then, by Lemma 9.12 part 1, we get $r^X_i = l^Y_q = i = q$, and then since $Y \preceq X$ gives us $l^Y_i \leq l^Y_j \leq r^Y_i \leq r^X_i$, we would have $r^X_i = r^Y_i$ which is not the case for non-empty $S_i$.

If $(s_i, s_i + 1, \ldots, r^X_i - 1)$ have already been removed from $S_i$ then, since only one term can be removed in each $T_j$ iteration, and a term $k$ is only removed if $k + 1 \in T_j$, there must be some sequences $T_{j_1}, T_{j_2}, \ldots, T_{j_m}$ with $q < j_1 < j_2 < \ldots < j_m$ such that $s_i + k \in T_{j_k}$ for each $1 \leq k \leq m$, and $s_i + m = r^X_i$. Thus $r^X_i \in T_{j_m}$ and, since the smallest term of $T_{j_m}$ is $l^X_{j_m}$, we have $r^X_i \geq l^X_{j_m}$. Also, since $S_i$ is non-empty we have $i \leq r^Y_i < r^X_i$ so by Lemma 9.12 part 1 we have $r^X_i \neq l^X_{j_m}$, and so $l^X_{j_m} < r^X_i$. By Lemma 9.12 part 3, the terms of $(l^X_1, l^X_2, \ldots, l^X_n)$ corresponding to non-empty $T_j$ form a strictly increasing sequence. Thus $l^X_q < l^X_{j_1} < l^X_{j_2} < \ldots < l^X_{j_m} < r^X_i$, and so $l^X_q < r^X_i - m = s_i$.

In either case we have shown that $s_i \neq l^X_q$ immediately before the $T_q$ iteration.

The proof of the second part is analogous to the first.

6) Let, at some point in the algorithm, $B = (b_1, b_2, \ldots, b_m)$ be a scattered subsequence of $S_n S_{n-1} \cdots S_1$ consisting of consecutive decreasing integers, and let column $b_1 + 1$ of $Y$ contain a lower corner. We know there exists $V \in DC_n$ such that $X = YV$. Let $\mathcal{Y}$ and $\mathcal{V}$ be sequences such that $\varepsilon_{\mathcal{Y}} = Y$ and $\varepsilon_{\mathcal{V}} = V$. We will first show that $B$ must be a scattered subsequence of $\mathcal{V}$. 
Since the $S_i$ sequences are increasing, each term of $B$ must lie in a different $S_i$. Let $i_1 > i_2 > \ldots > i_m$ be such that $b_j \in S_{i_j}$ for each $j \leq m$. For each $j$ the sequence $S_{i_j}$ is $(r_{i_j}^Y, r_{i_j}^Y + 1, \ldots, s_{i_j} - 1)$ and $s_{i_j} \leq r_{i_j}^X$ (since $s_{i_j}$ is initially defined as $r_{i_j}^X$ and never increased), so $r_{i_j}^Y \leq b_j \leq r_{i_j}^X - 1$.

Since $x_{i_j, r_{i_j}^X} = 1$, Lemma 9.8 tells us that $(i_j, i_j + 1, \ldots, r_{i_j}^X - 1)$ must be a scattered subsequence of $\mathcal{YV}$. Since $r_{i_j}^Y$ is the right-most 1 in row $r_{i_j}$ of $Y$, either $y_{i_j, r_{i_j}^Y + 1} = 0$ or $r_{i_j}^Y = n$. Either way $(i_j, i_j + 1, \ldots, r_{i_j}^Y)$ is not a scattered subsequence of $\mathcal{Y}$, and so $(r_{i_j}^Y, r_{i_j}^Y + 1, \ldots, r_{i_j}^X - 1)$ must be a scattered subsequence of $\mathcal{V}$, and $b_j$ is in this scattered subsequence.

Let $j < m$. We know that $(r_{i_j}^Y, r_{i_j}^Y + 1, \ldots, r_{i_j}^X - 1)$ and $(r_{i_{j+1}}^Y, r_{i_{j+1}}^Y + 1, \ldots, r_{i_{j+1}}^X - 1)$ are both scattered subsequences of $\mathcal{V}$. We also know that $b_j$ is a term of the first sequence, and $b_{j+1} = b_j - 1$ is a term of the second sequence. Assume for contradiction that these two scattered subsequences occur in $\mathcal{V}$ in such a way that $b_{j+1}$ from the second scattered subsequence occurs before $b_j$ from the first scattered subsequence. Then $(r_{i_{j+1}}^Y, r_{i_{j+1}}^Y + 1, \ldots, b_{j+1}, b_j, \ldots, r_{i_j}^X - 1)$ is a scattered subsequence of $\mathcal{V}$ consisting of consecutive increasing integers, and thus $(i_{j+1}, i_{j+1} + 1, \ldots, r_{i_j}^X - 1)$ is a scattered subsequence of $\mathcal{YV}$. Now by Lemma 9.8 position $(i_{j+1}, r_{i_j}^Y)$ of $YV$ is equal to 1. However, $S_{i_j}$ and $S_{i_{j+1}}$ are both non-empty and $i_j > i_{j+1}$, so Lemma 9.12 part 3 tells us that $r_{i_j}^X > r_{i_{j+1}}^X$. Thus position $(i_{j+1}, r_{i_j}^X)$ of $X$ is equal to 0, contradicting $YV = X$.

Therefore the two scattered subsequences of $\mathcal{V}$ are such that $b_j$ occurs before $b_{j+1}$. This holds for all $j < m$, so $B$ is a scattered subsequence of $\mathcal{Y}$.

By assumption, column $b_1 + 1$ of $Y$ contains a lower corner. Let $(j, b_1 + 1)$ be this lower corner. Then $y_{j, b_1 + 1} = 1$, so $(j - 1, j - 2, \ldots, b_1 + 1)$ is a scattered subsequence of $\mathcal{Y}$, and so $(j - 1, j - 2, \ldots, b_m)$ is a scattered subsequence of
Since $YV = X$ we get $x_{j:b_m} = 1$, and so $l_j^X \leq b_m$. Since $(j, b_1 + 1)$ is a lower corner of $Y$ we have $l_j^Y = b_1 + 1$. The sequence $T_j$ is initially defined as $(l_j^Y - 1, l_j^Y - 2, \ldots, l_j^X)$, and we now see that $B$ is a subsequence of this.

The proof of the second part is analogous to the first.

\[
\Box
\]

**Theorem 9.13**

Given $X, Y \in DC_n$, with the variables defined by the above algorithm and with $S_i, T_i, V$ and $V$ defined as above, if $Y$ is a left factor of $X$, then $X = YV$.

**Proof.** Let $X, Y \in DC_n$ be such that $Y$ is a left factor of $X$, and let each $l_i^X, r_i^X, l_i^Y, r_i^Y, s_i$ and $t_j$ be defined as in the algorithm, and each $S_i$ and $T_i$ be defined as above:

\[
S_i = \begin{cases} (r_i^Y, r_i^Y + 1, \ldots, s_i - 1) & \text{if } s_i > 0, \\ ( ) & \text{if } s_i = 0, \end{cases}
\]

\[
T_i = \begin{cases} (t_i - 1, t_i - 2, \ldots, l_i^X) & \text{if } t_i > 0, \\ ( ) & \text{if } t_i = 0. \end{cases}
\]

For each $i \leq n$ we let $S_i'$ and $T_i'$ be equal to the initial definitions of $S_i$ and $T_i$:

\[
S_i' = \begin{cases} (r_i^Y, r_i^Y + 1, \ldots, r_i^X - 1) & \text{if } r_i^X > r_{i-1}^X \text{ or } i = 1, \\ ( ) & \text{otherwise,} \end{cases}
\]

\[
T_i' = \begin{cases} (l_i^Y - 1, l_i^Y - 2, \ldots, l_i^X) & \text{if } l_i^X < l_{i+1}^X \text{ or } i = n, \\ ( ) & \text{otherwise.} \end{cases}
\]

We also define

\[
A = \varepsilon S_n S_{n-1} \cdots S_1, \quad A' = \varepsilon S_n S_n' \cdots S_1',
\]

\[
D = \varepsilon T_1 T_2 \cdots T_n, \quad D' = \varepsilon T_1 T_2' \cdots T_n'.
\]
so that $A$ is the matrix product corresponding to the $S_i$ sequences at the end of the algorithm, $D$ is the matrix product corresponding to the $T_j$ sequences at the end of the algorithm, and $A'$ and $D'$ are the same matrix products but taken in the middle of the algorithm, after line 12, when the sequences have been defined but not yet modified. The matrix $V$, as defined earlier, is equal to $AD$, so we need to show that $X = YAD$.

**Example 9.11 (continued)**

Recall our running example:

$$X = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}, \quad Y = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}.$$

In this example the $S_i$ and $T_j$ sequences are

$S_1 = (2, 3, 4), \quad S_7 = (9),$

$T_4 = (1), \quad T_5 = (2), \quad T_{10} = (9, 8, 7),$

$S_6' = (8), \quad S_7' = (9),$

$T_4' = (2, 1), \quad T_6' = (3, 2), \quad T_{10}' = (9, 8, 7),$

and the corresponding matrices are
It can be verified that $X = YAD$ in this case.

We split the proof of $X = YAD$ into a series of claims. Recall that the maps $U$ and $L$ map into $C_+^n$ and $C_-^n$, whose elements can be regarded as either matrices or functions. In these claims we regard the images of $X$ and $YAD$ under $U$ and $L$ as functions, and prove their equality as functions in order to prove that the matrices $X$ and $YAD$ are equal.

Claim 1. $U(X) = U(YA')$.

Claim 2. $L(X) = L(YD')$.

Claim 3. $U(YA') = U(YAD)$.

Claim 4. $L(YD') = L(YAD)$.

From Claims 1 and 3 we have $U(X) = U(YAD)$, and from Claims 2 and 4 we have $L(X) = L(YAD)$. Thus by Lemma 9.4 we have $X = U(X) \otimes L(X) =$
$U(YAD) \otimes L(YAD) = YAD$ as required. We just need to prove the claims.

**Claim 1.** $U(X) = U(YA')$.

**Proof of claim:** Let $k \leq n$. Given $M \in DC_n$, the function $U(M)$ maps $k$ to $\max\{j : m_{k,j} = 1\}$. Thus

\[ [k]U(X) = r^X_k \quad \text{and} \quad [k]U(Y) = r^Y_k. \]

Assume for now that $k \in S'_i$ for some $i$ and let $i' = \max\{i : k \in S'_i\}$. Since $S'_i = (r^Y_{i'}, r^Y_{i'} + 1, \ldots, r^X_{i'} - 1)$, we see that $(k, k+1, \ldots, r^X_{i'} - 1)$ is a scattered subsequence of $S'_nS'_{n-1} \cdots S'_1$, and so by Lemma 9.8 position $(k, r^X_{i'})$ of $A'$ is equal to 1.

Since $i' = \max\{i : k \in S'_i\}$, the occurrence of $k$ in $S'_i$ is the first occurrence of $k$ in $S'_nS'_{n-1} \cdots S'_1$. By Lemma 9.12 Part 3 the sequence $(r^X_1, r^X_2, \ldots, r^X_n)$ is non-decreasing, so for any $i \leq i'$ we have $r^X_{i'} > r^X_{i} - 1 = \max S'_i$. Therefore $r^X_{i'}$ does not occur in $S'_i$ and so the last occurrence of $r^X_{i'}$ in $S'_nS'_{n-1} \cdots S'_1$ (if any) is before the first occurrence of $k$. Thus $(k, k+1, \ldots, r^X_{i'})$ is not a scattered subsequence of $S'_nS'_{n-1} \cdots S'_1$ and so by Lemma 9.8 position $(k, r^X_{i'} + 1)$ of $A'$ is not equal to 1 (it is either equal to 0 or outside of the matrix).

Therefore $\max\{j : a'_{k,j} = 1\} = r^X_{i'}$, and so $[k]U(A') = r^X_{i'} = r^X_{\max\{i : k \in S'_i\}}$. Then since $(r^X_1, r^X_2, \ldots, r^X_n)$ is a non-decreasing sequence, $[k]U(A') = \max\{r^X_i : k \in S'_i\}$.

On the other hand if $k \notin S'_i$ for all $i$ then $k$ does not occur in $S'_nS'_{n-1} \cdots S'_1$ so by Lemma 9.8 position $(k, k+1)$ of $A'$ is not equal to 1. Therefore $\max\{j : a'_{k,j} = 1\} = k$ and so $[k]U(A') = k$. Thus

\[ [k]U(A') = \begin{cases} 
\max\{r^X_i : k \in S'_i\} & \text{if } k \in S'_i \text{ for some } i, \\
k & \text{otherwise.}
\end{cases} \]
Example 9.11 (continued)

In our running example we have

\[
A' = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},
\]

so as a function we have

\[
U(A')
\]

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & \multicolumn{2}{c}{2} & \multicolumn{2}{c}{3} & \multicolumn{2}{c}{4} & \multicolumn{2}{c}{5} & \multicolumn{2}{c}{6} \\
1 & \multicolumn{7}{c}{7} & \multicolumn{2}{c}{8} & \multicolumn{2}{c}{9} \\
1 & \multicolumn{10}{c}{10}
\end{array}
\]

In this example the non-empty \( S'_i \) sequences are

\[
S'_1 = (2, 3, 4), \quad S'_6 = (8), \quad S'_7 = (9),
\]

and so we see that

\[
[k]U(A') = \begin{cases} 
\max\{r_x^k : k \in S'_i\} & \text{if } k \in S'_i \text{ for some } i, \\
  k & \text{otherwise,}
\end{cases}
\]

as expected.

If we multiply \( U(A') \) on the left by \( U(Y) \) to give \( U(YA') \) we get
which is equal to $U(X)$:

As a function, $U(YA')$ behaves as follows:

$$[k]U(YA') = [k]U(Y)U(A')$$

$$= [r^Y_k]U(A')$$

$$= \begin{cases} 
\max\{r^X_i : r^Y_i \in S_i'\} & \text{if } r^Y_k \in S_i' \text{ for some } i, \\
 r^Y_k & \text{otherwise}, 
\end{cases}$$

and we must show that $U(X)$ is the same function.

First we consider the case when $r^Y_k \notin S_i'$ for all $i$. In this case $S_k'$ must be empty, otherwise it would be equal to $(r^Y_k, r^Y_k + 1, \ldots, r^X_k - 1)$ and would therefore contain $r^Y_k$. Therefore by line 9 of the algorithm either $r^X_k \leq r^Y_k$ or both $k > 1$ and $r^X_k \leq r^X_{k-1}$.

Assume for contradiction that $r^X_k > r^Y_k$. Then $r^X_k \leq r^X_{k-1}$ so, since $(r^Y_1, r^Y_2, \ldots, r^Y_n)$ is a non-decreasing sequence, $r^Y_{k-1} \leq r^Y_k < r^X_k \leq r^X_{k-1}$. Then $S_k'$ must also be empty, otherwise it would contain $r^Y_k$. Now by line 9, since $r^Y_{k-1} < r^X_{k-1}$, we have both $k - 1 > 1$ and $r^X_{k-1} \leq r^X_{k-2}$. Then $r^Y_{k-2} \leq r^Y_{k-1} \leq r^Y_k < r^X_k \leq r^X_{k-1} \leq r^X_{k-2}$, so $S'_{k-2}$ must be empty and, since $r^Y_{k-2} < r^X_{k-2}$, both $k - 2 > 1$ and $r^X_{k-2} \leq r^X_{k-3}$.
Repeating this logic, we eventually arrive at \( k - (n - 1) > 1 \), so \( k > n \), a contradiction.

Therefore \( r_k^X \leq r_k^Y \). Since \( Y \) is a left factor of \( X \) we have \( Y \preceq X \) by monotonicity, and so \( r_k^X \geq r_k^Y \). Thus \([k]U(X) = r_k^X = r_k^Y = [k]U(YA')\).

In the other case, \( r_k^Y \in S'_i \) for some \( i \). Let \( i' = \max\{i : r_k^Y \in S'_i\} \). Then \( r_k^Y \geq \min S'_i = r_{i'}^Y \). Since \( S'_i \) is non-empty, by line 9 of the algorithm either \( r_{i'}^Y > r_{i'-1}^Y \) or \( i' = 1 \). If \( r_{i'}^Y > r_{i'-1}^Y \), then \( r_{i'}^Y > r_{i'-1}^Y \) so, since \((r_1^Y, r_2^Y, \ldots, r_n^Y)\) is a non-decreasing sequence, \( k > i' - 1 \), and we have \( k \geq i' \). If \( i' = 1 \), then \( k \geq i' \) anyway. Therefore, since \((r_1^X, r_2^X, \ldots, r_n^X)\) is a non-decreasing sequence we have \( r_k^X \geq r_{i'}^X \). Then \( r_k^Y \leq \max S'_i = r_{i'}^X - 1 \leq r_k^X - 1 \), so \( r_k^Y < r_k^X \).

Let \( p = \min\{i : r_i^X = r_k^X\} \). Clearly \( p \leq k \), so \( r_p^Y \leq r_k^Y < r_k^X = r_p^X \). Since \( r_p^X \neq r_{p-1}^X \) by the definition of \( p \), there is an upper corner in row \( p \) of \( X \), so by line 9 of the algorithm and the fact that \( r_p^Y < r_p^X \), we see that \( S'_p \) is non-empty and is therefore equal to \((r_p^Y, r_p^Y + 1, \ldots, r_p^X - 1)\). Since \( r_p^Y \leq r_k^Y < r_p^X \) we have \( r_k^Y \in S'_p \), and so \( i' \geq p \). We have already seen that \( i' \leq k \), so \( p \leq i' \leq k \). By the definition of \( p \), the only \( i \) between \( p \) and \( k \) which satisfies \( r_i^X > r_{i-1}^X \) is \( p \), so the only one of the sequences \( S'_p, S'_{p+1}, \ldots, S'_k \) which is non-empty is \( S'_p \). Therefore \( i' = p \), and \([k]U(X) = r_k^X = r_p^X = r_{i'}^X = \max\{r_i^X : r_k^Y \in S'_i\} = [k]U(YA')\).

Thus, for each \( k \leq n \), in either case we have \([k]U(X) = [k]U(YA')\). Therefore \( U(X) = U(YA') \) as required.

Claim 2. \( L(X) = L(YD') \).

Proof of claim: Let \( k \leq n \). Given \( M \in DC_n \), the function \( L(M) \) maps \( k \) to \( \min\{j : m_{k,j} = 1\} \). Thus

\[
[k]L(X) = l_k^X \quad \text{and} \quad [k]L(Y) = l_k^Y.
\]
Assume for now that $k - 1 \in T_j'$ for some $j$ and let $j' = \min \{ j : k - 1 \in T_j' \}$. Since $T_j' = (l_j^Y - 1, l_j^Y - 2, \ldots, l_j^X)$, we see that $(k - 1, k - 2, \ldots, l_j^X)$ is a scattered subsequence of $T_1' T_2' \cdots T_n'$, and so by Lemma 9.8 position $(k, l_j^X)$ of $D'$ is equal to 1.

Since $j' = \min \{ j : k - 1 \in T_j' \}$, the occurrence of $k - 1$ in $T_j'$ is the first occurrence of $k - 1$ in $T_1' T_2' \cdots T_n'$. By Lemma 9.12 Part 3 the sequence $(l_1^X, l_2^X, \ldots, l_n^X)$ is non-decreasing, so for any $j \geq j'$ we have $l_j^X - 1 < l_j^X = \min T_j'$. Therefore $l_j^X - 1$ does not occur in $T_j'$, and the last occurrence of $l_j^X - 1$ in $T_1' T_2' \cdots T_n'$ (if any) is before the first occurrence of $k - 1$. Thus $(k - 1, k - 2, \ldots, l_j^X - 1)$ is not a scattered subsequence of $T_1' T_2' \cdots T_n'$ and so by Lemma 9.8 position $(k, l_j^X - 1)$ of $D'$ is not equal to 1 (it is either equal to 0 or outside of the matrix).

Therefore $\min \{ j : d'_{k,j} = 1 \} = l_j^X$, and so $[k] L(D') = l_j^X = l_{\min \{ j : k - 1 \in T_j' \}}$. Then since $(l_1^X, l_2^X, \ldots, l_n^X)$ is a non-decreasing sequence, $[k] L(D') = \min \{ l_j^X : k - 1 \in T_j' \}$.

On the other hand if $k - 1 \notin T_j'$ for all $j$ then $k - 1$ does not occur in $T_1' T_2' \cdots T_n'$ so by Lemma 9.8 position $(k, k - 1)$ of $D'$ is not equal to 1. Therefore $\min \{ j : d'_{k,j} = 1 \} = k$ and so $[k] L(D') = k$. Thus

$$[k] L(D') = \begin{cases} \min \{ l_j^X : k - 1 \in T_j' \} & \text{if } k - 1 \in T_j' \text{ for some } j, \\ k & \text{otherwise.} \end{cases}$$

**Example 9.11 (continued)**

In our running example we have
so as a function we have

\[ L(D') = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
& & & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
& & & & & & & & & \\
0 & 0 & 0 & 0 & & & & & & 1 \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & \\
& & & & & & & & & 1 & 1 & 1 & 0 \\
& & & & & & & & & & & & & & 1 & 1 & 1 & 1 \\
& & & & & & & & & & & & & & & & 1 & 1 & 1 & 1
\end{bmatrix}, \]

In this example the non-empty \( T_j' \) sequences are

\[ T_4' = (2, 1), \quad T_5' = (3, 2), \quad T_{10}' = (9, 8, 7), \]

and so we see that

\[ [k]L(D') = \begin{cases}
\min\{l^X_j : k - 1 \in T_j'\} & \text{if } k - 1 \in T_j' \text{ for some } j, \\
k & \text{otherwise},
\end{cases} \]

as expected.

If we multiply \( L(D') \) on the left by \( L(Y) \) to give \( L(YD') \) we get
which is equal to $L(X)$:

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{array}
\]

As a function, $L(YD')$ behaves as follows:

\[
[k]L(YD') = [k]L(Y)L(D')
\]

\[
= [l^Y_k]L(D')
\]

\[
= \begin{cases} 
\min\{l^X_j : l^Y_j - 1 \in T'_j\} & \text{if } l^Y_k - 1 \in T'_j \text{ for some } j, \\
l^Y_k & \text{otherwise,}
\end{cases}
\]

and we must show that $L(X)$ is the same function.

First we consider the case when $l^Y_k - 1 \notin T'_j$ for all $j$. In this case $T'_k$ must be empty, otherwise it would be equal to $(l^Y_k - 1, l^Y_k - 2, \ldots , l^X_k)$ and would therefore contain $l^Y_k - 1$. Therefore by line 11 of the algorithm either $l^X_k \geq l^Y_k$ or both $k < n$ and $l^X_k \geq l^X_{k+1}$.

Assume for contradiction that $l^X_k < l^Y_k$. Then $l^X_k \geq l^X_{k+1}$ so, since $(l^Y_1, l^Y_2, \ldots , l^Y_n)$ is a non-decreasing sequence, $l^Y_{k+1} \geq l^X_k > l^X_k \geq l^X_{k+1}$. Then $T'_{k+1}$ must also be empty, otherwise it would contain $l^Y_k - 1$. Now by line 11, since $l^Y_{k+1} > l^X_{k+1}$, we have both $k + 1 < n$ and $l^X_{k+1} \geq l^X_{k+2}$. Then $l^Y_{k+2} \geq l^Y_{k+1} \geq l^Y_k > l^X_k \geq l^X_{k+1} \geq l^X_{k+2}$, so $T'_{k+2}$ must be empty and, since $l^Y_{k+2} > l^X_{k+2}$, both $k + 2 < n$ and $l^X_{k+2} \geq l^X_{k+3}$. Repeating this logic, we eventually arrive at $k + (n - 1) < n$, so $k < 1$, a contradiction.

Therefore $l^X_k \geq l^Y_k$. Since $Y$ is a left factor of $X$ we have $Y \preceq X$ by monotonicity, and so $l^X_k \leq l^Y_k$. Thus $[k]L(X) = l^X_k = l^Y_k = [k]L(YD')$.

In the other case, $l^Y_k - 1 \in T'_j$ for some $j$. Let $j' = \min\{j : l^Y_k - 1 \in T'_j\}$. Then $l^Y_k - 1 \leq \max T'_{j'} = l^Y_{j'} - 1$, so $l^Y_k \leq l^Y_{j'}$. Since $T'_{j'}$ is non-empty, by line 11
of the algorithm either \( l'_{j'} < l'_{j'+1} \) or \( j' = n \). If \( l'_{j'} < l'_{j'+1} \) then \( l'_{k} < l'_{j'+1} \) so, since 
(\( l'_{1}, l'_{2}, \ldots , l'_{n} \)) is a non-decreasing sequence, \( k < j' + 1 \), and we have \( k \leq j' \). If \( j' = n \) then \( k \leq j' \) anyway. Therefore, since 
(\( l_{1}^{X}, l_{2}^{X}, \ldots , l_{n}^{X} \)) is a non-decreasing sequence we have \( l_{k}^{X} \leq l_{j'}^{X} \). Then \( l_{k}^{X} - 1 \geq \min \{T_{j'} = l_{j'}^{X} \geq l_{k}^{X}, \) so \( l_{k}^{X} > l_{k}^{X} \).

Let \( p = \max \{ j : l_{j}^{X} = l_{k}^{X} \} \). Clearly \( p \geq k \), so \( l_{p}^{X} \geq l_{k}^{X} > l_{k}^{X} = l_{p}^{X} \). Since 
\( l_{p+1}^{X} \neq l_{p}^{X} \) by the definition of \( p \), there is a lower corner in row \( p \) of \( X \), so by line 11 of the algorithm and the fact that \( l_{p}^{X} > l_{p}^{X} \), we see that \( T_{p} \) is non-empty and is therefore equal to \( (l_{p}^{X} - 1, l_{p}^{X} - 2, \ldots , l_{p}^{X}) \). Since \( l_{p}^{X} \geq l_{k}^{X} \) we have \( l_{k}^{X} - 1 \in T_{p} \), and so \( j' \leq p \). We have already seen that \( k \leq j' \), so \( k \leq j' \leq p \). By the definition of \( p \), the only \( j \) between \( k \) and \( p \) which satisfies \( l_{j+1}^{X} > l_{j}^{X} \) is \( p \), so the only one of the sequences \( T_{k}, T_{k+1}, \ldots , T_{p} \) which is non-empty is \( T_{p} \). Therefore \( j' = p \), and 
\[ [k]L(X) = l_{k}^{X} = l_{p}^{X} = l_{p}^{X} = \min \{l_{j}^{X} : l_{k}^{X} - 1 \in T_{j} \} = [k]L(YD') \]

Thus, for each \( k \leq n \), in either case we have \([k]L(X) = [k]L(YD') \). Therefore 
\( L(X) = L(YD') \) as required. \( \blacksquare \)

**Claim 3.** \( U(YA') = U(YAD) \).

**Proof of claim:** We wish to prove that \( U(YA') = U(YAD) \), or equivalently 
\( U(Y)U(A') = U(Y)U(AD) \), so we need to show that \([k]U(A') = [k]U(AD) \) for each \( k \) in the image of \( U(Y) \).

The differences between \( A' \) and \( A \) are due to terms being removed from the 
\( S \) sequences. Terms are removed from the \( S \) sequences only in the \( T_{q} \) iterations, 
so we consider what happens on each of these iterations. For a fixed \( q \) such that 
\( T_{q} \) is non-empty, Lemma 9.12 part 4 tells us that no terms are removed from \( T_{q} \) 
after the \( T_{q} \) iteration, so \( T_{q} \) is the same during its own iteration as it is at the 
end of the algorithm. For this claim we let each \( S \) be as it is immediately before 
the \( T_{q} \) iteration. Let \( A^{[q]} \) be equal to \( A = \varepsilon S_{n}s_{n-1} \cdots S_{i} \) and let \( A^{[q+]} \) be the same 
matrix product defined on the values of \( S \) as they will be immediately after the \( T_{q} \) 
iteration.
If \( w_1 > w_2 > \ldots > w_m \) index the non-empty \( \mathcal{T}_j \) sequences in decreasing order (the same order in which the \( \mathcal{T}_j \) iterations occur in the algorithm), then \( A^{[w_1]} = A' \) and \( A^{[w_m]} = A \), and for each \( k < m \) we have \( A^{[w_k+]} = A^{[w_{k+1}]} \). Given \( q \leq n \) such that \( \mathcal{T}_q \) is non-empty we will show that, as functions, \( U(A^{[q]}) = U(A^{[q+1]})U(\varepsilon_{\mathcal{T}_q}) \) over the image of \( U(Y) \). Then

\[
U(YA') = U(Y)U(A^{[w_1]}) \\
= U(Y)U(A^{[w_1+1]})U(\varepsilon_{\mathcal{T}_{w_1}}) \\
= U(Y)U(A^{[w_2]})U(\varepsilon_{\mathcal{T}_{w_1}}) \\
= U(Y)U(A^{[w_2+1]})U(\varepsilon_{\mathcal{T}_{w_2}})U(\varepsilon_{\mathcal{T}_{w_1}}) \\
= U(Y)U(A^{[w_3]})U(\varepsilon_{\mathcal{T}_{w_2} \mathcal{T}_{w_1}}) \\
\vdots \\
= U(Y)U(A^{[w_m+1]})U(\varepsilon_{\mathcal{T}_{w_m} \mathcal{T}_{w_{m-1}} \mathcal{T}_{w_2} \mathcal{T}_{w_1}}) \\
= U(YA\varepsilon_{\mathcal{T}_{m} \mathcal{T}_{2} \mathcal{T}}) \\
= U(YAD),
\]

as required.

**Example 9.11 (continued)**

We will look at our example during the \( \mathcal{T}_{10} \) iteration.

We have \( \mathcal{T}_{10} = (9, 8, 7) \), so \( \varepsilon_{\mathcal{T}_{10}} \) is equal to

\[
\varepsilon_{\mathcal{T}_{10}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{bmatrix},
\]
and the function \( U(\varepsilon_{T_{10}}) \) is

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
2 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
3 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
4 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
5 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
6 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
7 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
8 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
9 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
10 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
\end{array}
\]

Immediately before the \( T_{10} \) iteration the only non-empty \( S_i \) sequences are

\[ S_1 = (2, 3, 4), \quad S_6 = (8), \quad S_7 = (9), \]

so the matrix \( A^{[10]} \) is

\[
A^{[10]} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix},
\]

and the function \( U(A^{[10]}) \) is

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
2 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
3 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
4 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
5 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
6 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
7 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
8 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
9 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
10 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
\end{array}
\]

Immediately after the \( T_{10} \) iteration the only non-empty \( S_i \) sequences are

\[ S_1 = (2, 3, 4), \quad S_7 = (9), \]

so the matrix \( A^{[10+]} \) is
and the function $U(A^{[10+]})$ is

$$
U(A^{[10+]}) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix},
$$

Now $U(Y)U(A^{[10+]}) = U(Y)U(A^{[10+]})U(\varepsilon_{T_0})$ as follows:
Given $q \leq n$ such that $T_q$ is non-empty we need to show that $U(A^{[q]}) = U(A^{[q+1]})U(\varepsilon_{T_q})$ over the image of $U(Y)$. We consider the effect of $U(A^{[q]})$ and $U(A^{[q+1]})$ as functions. Recall that we are using the values of the $S_i$ sequences immediately before the $T_q$ iteration. Choose some $k \leq n$, and if $k \in S_i$ for some $i$ let $i' = \max\{i : k \in S_i\}$ and $l = s_{i'}$. Then, as in the proof of Claim 1,

$$[k]U(A^{[q]}) = \begin{cases} l & \text{if } k \in S_i \text{ for some } i, \\ k & \text{if } k \notin S_i \text{ for all } i. \end{cases}$$

$U(A^{[q+1]})$ is the same function, but using the values of the $S_i$ sequences after the $T_q$ iteration.

If we have $k \notin S_i$ for all $i$ before the $T_q$ iteration then the same will still be true after the $T_q$ iteration, and so $U(A^{[q+1]})$ maps $k$ to $k$. Otherwise we have defined $i' = \max\{i : k \in S_i\}$ and $l = s_{i'}$. Since the algorithm only removes terms from the sequences and never adds them, $k$ will never be a term of $S_i$ for any $i > i'$. Thus, unless the $T_q$ iteration removes $k$ from $S_{i'}$, we will still have $i' = \max\{i : k \in S_i\}$ after the $T_q$ iteration. Therefore:

- if the $T_q$ iteration does not remove any terms from $S_{i'}$ then $U(A^{[q+1]})$ maps $k$ to the same value as $U(A^{[q]})$ does, which is $l$,

- if the $T_q$ iteration does remove a term from $S_{i'}$ but this term is not $k$, then $U(A^{[q+1]})$ maps $k$ to $l - 1$,

- if the $T_q$ iteration removes $k$ from $S_{i'}$ then, immediately before it is removed, $k$ must be the largest term of $S_{i'}$, so we have $k = s_{i'} - 1$. Since, by Lemma 9.12 Part 3, the non-zero terms of $(s_1, s_2, \ldots, s_n)$ form a non-decreasing sequence, after the $T_q$ iteration $k$ is not a term of $S_i$ for any $i \leq i'$, and is therefore not a term of any $S_i$ at all. Thus $U(A^{[q+1]})$ maps $k$ to $k = s_{i'} - 1 = l - 1$. 


Therefore the action of $U(A^{[q+1]})$ as a function is

$$[k]U(A^{[q+1]}) = \begin{cases} 
l & \text{if } k \in S_i \text{ for some } i \text{ and } l \notin T_q, \\
l - 1 & \text{if } k \in S_i \text{ for some } i \text{ and } l \in T_q, \\
k & \text{if } k \notin S_i \text{ for all } i. 
\end{cases}$$

Since $T_q$ is a descending sequence, by Lemma 9.8, $\max\{j : (\varepsilon_T)_k,j = 1\}$ is $k + 1$ if $k \in T_q$ and $k$ if $k \notin T_q$. Thus, the action of $U(\varepsilon_{T_q})$ as a function is

$$[k]U(\varepsilon_{T_q}) = \begin{cases} 
k + 1 & \text{if } k \in T_q, \\
k & \text{if } k \notin T_q. 
\end{cases}$$

If $k \in S_i$ for some $i$ and $l \notin T_q$ then

$$[k]U(A^{[q+1]})U(\varepsilon_{T_q}) = [l]U(\varepsilon_{T_q})$$

$$= l$$

$$= [k]U(A^{[q]}).$$

If $k \in S_i$ for some $i$ and $l \in T_q$ then $l = s_i$ for some $i$ by the definition of $l$, and so by Lemma 9.12 part 5, $l - 1$ must also be a term of $T_q$. Thus

$$[k]U(A^{[q+1]})U(\varepsilon_{T_q}) = [l - 1]U(\varepsilon_{T_q})$$

$$= l$$

$$= [k]U(A^{[q]}).$$

If $k \notin S_i$ for all $i$ and $k \notin T_q$ then

$$[k]U(A^{[q+1]})U(\varepsilon_{T_q}) = [k]U(\varepsilon_{T_q})$$

$$= k$$

$$= [k]U(A^{[q]}).$$
The only remaining case is \( k \notin S_i \) for all \( i \), and \( k \in T_q \). We will show that in this case \( k \notin \text{Im}(U(Y)) \). Assume for contradiction that \( k \in \text{Im}(U(Y)) \).

Let \( i = \min \{ i' : [i']U(Y) = k \} \). Then \( k = [i]U(Y) = r_i^Y \), and since \( U(Y) \) is non-decreasing we have \( i \leq k \). Position \( (i, r_i^Y) \) of \( Y \) is equal to 1, but by the definitions of \( r_i^Y \) and \( i \) there is not a 1 to the right or above position \( (i, r_i^Y) \).

Therefore column \( r_i^Y = k \) of \( Y \) contains an upper corner. Since \( k \in T_q \) and column \( k \) of \( Y \) contains an upper corner, Lemma 9.12 part 6 tells us that \( k \in S_i' \) for some \( i' \), and so \( \varepsilon_k \) is a factor of \( A' \). Then since \( y_{i,k} = 1 \) and \( \varepsilon_k \) is a factor of \( A' \) we have \( (YA')_{i,k+1} = 1 \). By Claim 1 we have \( U(X) = U(YA') \), so \( X \) and \( YA' \) are equal above the diagonal and, since \( k \geq i \), we have \( x_{i,k+1} = 1 \). It follows that \( k + 1 \leq r_i^X \).

Let \( p = \min \{ i' : r_i^{X'} = r_i^X \} \). Then \( p \leq i \) so \( r_p^{Y'} \leq r_p^Y < k + 1 \leq r_i^X = r_p^X \).

\( S_p \) was initially defined as \( (r_p^{Y'}, r_p^{Y'} + 1, \ldots, r_p^{X'} - 1) \), and we have \( r_p^Y \leq k < r_p^X \), so \( k \) was initially in \( S_p \). Immediately before the \( T_q \) iteration we have \( k \notin S_p \), so \( k \) must have been removed from \( S_p \) by the \( T_j \) iteration for some \( j > q \). Therefore \( k + 1 \in T_j \) and so \( (k, k+1) \) is a scattered subsequence of \( T_qT_j \). Now Lemma 9.12 part 6 tells us that \( (k, k+1) \) is a subsequence of \( S_{i'}' \) for some \( i' \), so \( \varepsilon_k \) and \( \varepsilon_{k+1} \) are factors of \( A' \) and appear in that order. Then since \( y_{i,k} = 1 \) we get \( (YA')_{i,k+2} = 1 \), and since \( U(X) = U(YA') \) we get \( x_{i,k+2} = 1 \). Thus \( k + 2 \leq r_i^X \).

Now \( r_p^Y \leq k + 1 < r_i^X = r_p^X \), so \( k + 1 \) was initially in \( S_p \). Each iteration of the while loop can remove at most one term from \( S_p \), so \( k + 1 \) must have been removed before \( k \) was. Thus there is some \( j' > j \) such that \( k + 2 \in T_{j'} \), and by the same reasoning as above, \( k + 2 \) is initially in \( S_p \). By repeating this argument we eventually see that \( n \) is initially in \( S_p \), which is the required contradiction since the highest term initially in \( S_p \) is \( r_p^X - 1 \leq n - 1 \).

\[ \text{Claim 4. } L(YD') = L(YAD) \]

\[ \text{Proof of claim: } \] For this claim note that since elements of \( C_n^+ \) are upper triangular matrices and elements of \( C_n^- \) are lower triangular matrices, the transposition
function is a map from $C_n^+$ to $C_n^-$ and back again. We define a function $T : DC_n \rightarrow C_n^+$ by letting $T(A) = L(A)^T$. Since transposition reverses the order of multiplication, for this claim we will compose elements of $C_n^+$ from right to left and write the operand on the right so that $T(AB)[k] = T(A)T(B)[k] = T(A)[T(B)[k]]$. For a matrix $M \in DC_n$ the function $T(M)$ maps $k$ to $\max \{i : m_{i,k} = 1\}$.

We wish to prove that $L(YD') = L(YAD)$, or equivalently $T(YD') = T(YAD)$. For an arbitrary function $f : A \rightarrow B$ we define an equivalence relation on $A$ by $x = f y \iff f(x) = f(y)$. We therefore wish to show that $T(D') = T(\hat{Y}) T(AD)$ (by which we mean $T(D')[k] = T(\hat{Y}) T(AD)[k]$ for all $k \leq n$).

The differences between $D'$ and $D$ are due to terms being removed from the $T_j$ sequences. Terms are removed from the $T_j$ sequences only in the $S_p$ iterations, so we consider what happens in each of these iterations. For a fixed $p$ such that $S_p$ is non-empty, Lemma 9.12 part 4 tells us that no terms are removed from $S_p$ after the $S_p$ iteration, so $S_p$ is the same during its own iteration as it is at the end of the algorithm. For this claim we let each $T_j$ be as it is immediately before the $S_p$ iteration. Let $D_{[p]}$ be equal to $D = \varepsilon T_1 T_2 \cdots T_n$, and let $D_{[p+1]}$ be the same matrix product defined on the values of $T_j$ as they will be immediately after the $S_p$ iteration. Let $\hat{A}_p = \varepsilon S_n S_{n-1} \cdots S_{p+1}$.

If $w_1 > w_2 > \ldots > w_m$ index the non-empty $S_i$ sequences in decreasing order (the same order in which the $S_i$ iterations occur in the algorithm), then $D_{[w_1]} = D'$ and $D_{[w_{m+1}]} = D$, and for each $k < m$ we have $D_{[w_k+]} = D_{[w_{k+1}]}$. Also $\hat{A}_{w_1} = I_n$ as it corresponds to the empty sequence, $\hat{A}_{w_m} \varepsilon S_{w_m} = A$, and for each $k < m$ we have $\hat{A}_{w_k} \varepsilon S_{w_k} = \hat{A}_{w_{k+1}}$.

Given $p \leq n$ such that $S_p$ is non-empty we will show that $T(D_{[p]}) = T(Y\hat{A}_p)$.
$T(\varepsilon_{S_p})T(D^{[p+1]})$. Then

\[
T(YD') = T(Y\hat{A}_{w_1})T(D^{[w_1]})
\]

\[
= T(Y\hat{A}_{w_1})T(\varepsilon_{S_{w_1}})T(D^{[w_1+1]})
\]

\[
= T(Y\hat{A}_{w_2})T(D^{[w_2]})
\]

\[
= T(Y\hat{A}_{w_2})T(\varepsilon_{S_{w_2}})T(D^{[w_2+1]})
\]

\[
= T(Y\hat{A}_{w_3})T(D^{[w_3]})
\]

\[
= T(Y\hat{A}_{w_3})T(\varepsilon_{S_{w_3}})T(D^{[w_3+1]})
\]

\[\vdots\]

\[
= T(Y\hat{A}_{w_m})T(\varepsilon_{S_{w_m}})T(D^{[w_m+1]})
\]

\[
= T(YAD),
\]

as required.

**Example 9.11 (continued)**

We will look at our example during the $S_1$ iteration.

We have $S_1 = (2, 3, 4)$, so $\varepsilon_{S_1}$ is equal to

\[
\varepsilon_{S_1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix},
\]

and the function $T(\varepsilon_{S_1})$ is

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\end{array}
\]
Immediately before the $S_1$ iteration the only non-empty $T_j$ sequences are

$$T_4 = (2, 1), \quad T_5 = (3, 2), \quad T_{10} = (9, 8, 7),$$

so the matrix $D^{[1]}$ is

$$D^{[1]} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix},$$

and the function $T(D^{[1]})$ is

$$T(D^{[1]})$$

Immediately after the $S_1$ iteration the only non-empty $T_j$ sequences are

$$T_4 = (1), \quad T_5 = (2), \quad T_{10} = (9, 8, 7),$$

so the matrix $D^{[1+]}$ is

$$D^{[1+]} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix},$$

$$D^{[1+]} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix},$$
and the function $T(D^{[1+]})$ is

$$T(D^{[1+]})$$

The only non-empty $S_i$ with $i > p$ is $S_7 = (9)$. Thus the matrix $\hat{A}_1 = \varepsilon_{S_{10}S_9\cdots S_2}$ is equal to $\varepsilon_9$, and the function $T(\hat{A}_1)$ is

$$T(\hat{A}_1)$$

Now $T(Y)T(\hat{A}_1)T(D^{[1]}) = T(Y)T(\hat{A}_1)T(\varepsilon_{S_1})T(D^{[1+]})$ as follows (recall we are composing functions from right to left):
Given $p \leq n$ such that $\mathcal{S}_p$ is non-empty we need to show that $T(D^{[p]}) = T(Y_{\hat{A}_p}) T(\varepsilon_{\mathcal{S}_p}) T(D^{[p+1]})$. We consider the effect of $T(D^{[p]})$ and $T(D^{[p+1]})$ as functions. Recall that we are using the values of the $T_j$ sequences immediately before the $\mathcal{S}_p$ iteration. Choose some $k \leq n$, and if $k \in T_j$ for some $j$ let $l = \max\{t_j : k \in T_j\}$.

Taking the transpose reverses the order of multiplication, so, since the $\mathcal{S}_i$ sequences are ascending and the $T_j$ sequences are descending, the transposes of the matrices corresponding to the $T_j$ sequences (including those matrices which we also regard as functions) have the same kind of structure as the (untransposed) matrices corresponding to the $\mathcal{S}_i$ sequences. We can therefore use the same reasoning as in the proof of Claims 1 and 3 to see that

$$T(D^{[p]})[k] = \begin{cases} l & \text{if } k \in T_j \text{ for some } j, \\ k & \text{if } k \notin T_j \text{ for all } j, \end{cases}$$

and

$$T(D^{[p+1]})[k] = \begin{cases} l & \text{if } k \in T_j \text{ for some } j \text{ and } l \notin \mathcal{S}_p, \\ l - 1 & \text{if } k \in T_j \text{ for some } j \text{ and } l \in \mathcal{S}_p, \\ k & \text{if } k \notin T_j \text{ for all } j, \end{cases}$$
and
\[
T(\varepsilon_{\mathcal{S}_p})[k] = \begin{cases} 
  k + 1 & \text{if } k \in \mathcal{S}_p, \\
  k & \text{if } k \notin \mathcal{S}_p.
\end{cases}
\]

If \( k \in \mathcal{T}_j \) for some \( j \) and \( l \notin \mathcal{S}_p \) then
\[
T(\varepsilon_{\mathcal{S}_p})T(D^{[p+1]}[k] = T(\varepsilon_{\mathcal{S}_p})[l]
= l
= T(D^{[p]}[k]).
\]

If \( k \in \mathcal{T}_j \) for some \( j \) and \( l \in \mathcal{S}_p \) then \( l = t_j \) for some \( j \) by the definition of \( l \), and so by Lemma 9.12 part 5, \( l - 1 \) must also be a term of \( \mathcal{S}_p \). Thus
\[
T(\varepsilon_{\mathcal{S}_p})T(D^{[p+1]}[k] = T(\varepsilon_{\mathcal{S}_p})[l - 1]
= l
= T(D^{[p]}[k]).
\]

If \( k \notin \mathcal{T}_j \) for all \( j \) and \( k \notin \mathcal{S}_p \) then
\[
T(\varepsilon_{\mathcal{S}_p})T(D^{[p+1]}[k] = T(\varepsilon_{\mathcal{S}_p})[k]
= k
= T(D^{[p]}[k]).
\]

The only remaining case is \( k \notin \mathcal{T}_j \) for all \( j \), and \( k \in \mathcal{S}_p \). In this case
\[
T(\varepsilon_{\mathcal{S}_p})T(D^{[p+1]}[k] = T(\varepsilon_{\mathcal{S}_p})[k]
= k + 1
\neq k
= T(D^{[p]}[k]).
\]
We will show that $k = T(Y \hat{A}_p) k + 1$ by letting $k_1 = T(\hat{A}_p)[k]$ and $k_2 = T(\hat{A}_p)[k+1]$ and assuming for contradiction that $T(Y)[k_1] \neq T(Y)[k_2]$. Since $T(\hat{A}_p)$ is order-preserving we have $k_1 \leq k_2$, and since $T(Y)[k_1] = \max\{i : y_{i,k_1} = 1\}$ and $T(Y)[k_2] = \max\{i : y_{i,k_2} = 1\}$ the assumption means that columns $k_1$ and $k_2$ of $Y$ have their bottom-most 1s in different rows. Therefore there is a lower corner in column $j$ of $Y$ for some $k_1 < j \leq k_2$ such that $T(Y)[j] = T(Y)[k_2]$. Since $T(\hat{A}_p)$ is non-decreasing we have $k \leq k_1$, and so $k < j$.

Since $k_2 = T(\hat{A}_p)[k + 1]$ we have $(\hat{A}_p)_{k_2,k+1} = 1$ and so by Lemma 9.8 we must have $(k_2 - 1, k_2 - 2, \ldots, k + 1)$ as a scattered subsequence of $S_nS_{n-1} \cdots S_{p+1}$. We also have $k \in S_j$, so, since $k < j \leq k_2$ we find that $(j - 1, j - 2, \ldots, k)$ is a scattered subsequence of $S_nS_{n-1} \cdots S_p$. Then since there is a lower corner in column $j$ of $Y$, Lemma 9.12 part 6 tells us that $(j - 1, j - 2, \ldots, k)$ was a subsequence of some $T_{j'}$ when it was initially defined.

Since $k \notin T_{j'}$ immediately before the $S_p$ iteration it must have been removed before then. Each iteration of the while loop can remove at most one term from $T_{j'}$, and the terms are removed from the start of the sequence, so we must have $(j, j - 1, \ldots, k + 1)$ as a scattered subsequence of $S_nS_{n-1} \cdots S_{p+1}$ with each term being in a different $S_i$. Let $S_{i_j}, S_{i_{j-1}}, \ldots, S_{i_{k+1}}$ be such that $i_j > i_{j-1} > \ldots > i_{k+1}$ and $j' \in S_{i_{j'}}$ for each $j \geq j' \geq k + 1$.

If column $j$ of $Y$ is equal to a unit vector (necessarily the $j$th unit vector by reflexivity) then $T(Y)[j] = j$. By Lemma 6.7 row $j$ of $Y$ is also equal to the $j$th unit vector, so no value other than $j$ gets mapped to $j$ by $T(Y)$. Then since $T(Y)[j] = T(Y)[k_2]$ we must have $j = k_2$. Therefore $(k_2, k_2 - 1, \ldots, k + 1)$ is a scattered subsequence of $S_nS_{n-1} \cdots S_{p+1}$, and Lemma 9.8 gives us $(\hat{A}_p)_{k_2+1,k+1} = 1$, from which it follows that $T(\hat{A}_p)[k + 1] \geq k_2 + 1$, a contradiction.

On the other hand, if column $j$ of $Y$ is not equal to a unit vector then since there is a lower corner in this column there cannot also be an upper corner, by
Lemma 9.12 part 1. Thus, since $S_{ij}$ contains $j$, it must also contain $j - 1$. Since the first term of each $S_{ij}$ is $r^Y_{ij}$, and by Lemma 9.12 part 3 these terms form a strictly increasing sequence, we must have $j - 2 \in S_{ij}$, and in general for each $j \leq j' \leq k + 1$ we have $j' - 1 \in S_{ij'}$. Thus $(j - 1, j - 2, \ldots, k)$ is a scattered subsequence of $S_nS_{n-1} \cdots S_{p+1}$ so by Lemma 9.8 we have $(\hat{A})_{j,k} = 1$. Since $k_1 < j$ this contradicts $T(\hat{A})[k] = k_1$. 

This proves all four claims and the theorem follows.

**Theorem 9.14**

The decision problem DCMP is in P.

**Proof.** Given $C \in B_n$, DCMP asks if $C \in DC_n$. Let $X = C$ and $Y = I_n$ and follow the above algorithm. If $C \in DC_n$ then by Theorem 9.13 we have

$$C = YV = V.$$  

Conversely, if the above equation holds then $C \in DC_n$ since $V$ is a product of generators of $DC_n$. Thus we can find if $C \in DC_n$ by following the algorithm, calculating $V$, and checking if $C = V$, which takes polynomial time.

**Theorem 9.15**

The decision problems DCLP and DCRP are in P.

**Proof.** Given $X, Y \in DC_n$, DCLP asks if $X \leq_L Y$ and DCRP asks if $X \leq_R Y$.

Since $X \leq_L Y \iff X^T \leq_R Y^T$ and $DC_n^T = DC_n$ it suffices to show that DCRP is in P. We follow the above algorithm for $X$ and $Y$, and by Theorem 9.13 if $X \leq_R Y$ then we have $X = YV$. Conversely, if $X = YV$ then $X \leq_R Y$ by the definition of the $R$-order. Thus we can check if $X \leq_R Y$ by following the algorithm, calculating $V$, and checking if $X = YV$, which takes polynomial time.
Bibliography


