Monoids and the State Complexity of the Operation root(L)

by

Bryan Krawetz

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2003

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, we cover the general topic of state complexity. In particular, we examine the bounds on the state complexity of some different representations of regular languages. As well, we consider the state complexity of the operation root(L).

We give quick treatment of the deterministic state complexity bounds for nondeterministic finite automata and regular expressions. This includes an improvement on the worst-case lower bound for a regular expression, relative to its alphabetic length.

The focus of this thesis is the study of the increase in state complexity of a regular language L under the operation root(L). This operation requires us to examine the connections between abstract algebra and formal languages.

We present results, some original to this thesis, concerning the size of the largest monoid generated by two elements. Also, we give good bounds on the worst-case state complexity of root(L). In turn, these new results concerning root(L) allow us to improve previous bounds given for the state complexity of two-way deterministic finite automata.

Acknowledgements

First, I would like to thank Jeffrey Shallit for electing to supervise my studies, and for providing an environment where I was free to pursue whatever interested me. I would also like to thank Jeff for his ideas and guidance, which helped drive and shape this work.

Second, I would like to thank John Lawrence, for not only agreeing to read this thesis, but more importantly, for his work on the core problem, which helped me to gain valuable insight and understanding.

Third, I would like to thank Prabhakar Ragde for agreeing to read this thesis, and Narad Rampersad for helpful discussions and suggestions.

Finally, I would like to thank Erika and my family, who have supported me throughout my education.

Contents

1 Introduction **1**

2 State Complexity of Representations of Regular Languages **5**
2.1 Deterministic State Complexity of NFAs 7
2.2 Deterministic State Complexity of Regular Expressions 9

3 Monoids **15**
3.1 The Monoid of Transformations of a Finite Set 18
3.2 The Group of Permutations of a Finite Set 19
3.3 Monoids Generated by Finite Sets 20
3.3.1 Sets of Size Two: The Single Cycle Case 21
3.3.2 Sets of Size Two: The General Case 28

4 State complexity of root(L) **45**
4.1 Unary languages 49
4.2 Languages on larger alphabets 51
4.2.1 Constructing Automata from Monoids 52
4.2.2 Alphabets of Size Two 57
4.2.3 Alphabets of Size Three or More 61
4.3 An Application to the State Complexity of 2DFAs 61

5 Open Problems **67**

Bibliography **71**

List of Tables

1.2 State complexity of some basic operations 3

List of Figures

2.2 The NFA for $(0 + (01^*)^{n-1}0)^*$. 7
2.5 The NFA for $(0^*(01^*)^{r-1}0)^*$. 10

3.2 The automaton $\mathcal{M}$. 17
3.11 The graph G. 25

4.9 The automaton $\mathcal{A}_{\Phi,Y}$. 53

Chapter 1

Introduction

A *language* is a set of strings over some finite set of symbols called an *alphabet*. The set of all strings over the alphabet Σ is denoted by Σ^*; this includes the empty string, denoted by ϵ. In terms of the Chomsky hierarchy, a *class* of languages is a set of languages that can specified using a particular model of description.

One of the simplest models commonly used, in terms of descriptive capability, is the deterministic finite automaton, which is defined as follows.

A *deterministic finite automaton*, or *DFA*, is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite non-empty set of states, Σ is the finite input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. We assume that δ is defined on all elements of its domain. The domain of δ can be extended in the obvious way to $Q \times \Sigma^*$, where Σ^* is the free monoid over the alphabet Σ. For a DFA $\mathcal{A}$, the set $L(\mathcal{A}) = \{w \in \Sigma^* \ : \ \delta(q_0, w) \in F\}$ is said to be the language recognized by $\mathcal{A}$.

The class of languages recognized by DFAs is known as the class of *regular languages*. There are, however, a number of other models that can be used to describe this same class. Some of these other models include regular expressions, non-deterministic finite automata, and regular grammars. For a thorough introduction to languages and automata theory, consult a suitable reference text such as the one by Hopcroft and Ullman [11].

The *descriptional complexity* of a language is a measure of the amount of information required to represent a language using a particular model. One such measure, for regular languages, is state complexity. The *deterministic state complexity* (or simply *state com-*

plexity) of a language $L \subseteq \Sigma^*$, denoted by $\mathrm{sc}(L)$, is defined as the size (the number of states) of the smallest DFA recognizing L. Of course, there are many other measures of the descriptional complexity of regular languages; for a survey, see the work of Ellul [7].

In this thesis, we are concerned, almost exclusively, with the state complexity of regular languages. Naturally then, given a regular language, we need a method to determine the number of states in its minimal DFA. The technique we employ is based on the *Myhill–Nerode* equivalence relation, denoted $\sim_L$, defined as follows.

Let L be a regular language. Then for $x, y \in \Sigma^*$, $x \sim_L y$ when

$$xz \in L \Leftrightarrow yz \in L, \text{ for all } z \in \Sigma^*.$$

The equivalence relation $\sim_L$ is *right-invariant*, that is if $x \sim_L y$ then $xz \sim_L yz$ for all $z \in \Sigma^*$.

The following results, proven by Nerode [21], provide us with method for determining the size of the minimal DFA recognizing a language.

Theorem 1.1 (Nerode). *Let $L \subseteq \Sigma^*$ be a regular language. Then the number of states in the minimal DFA recognizing L is equal to the number of equivalence classes of the relation $\sim_L$.*

Note, however, that the actual statement of Theorem 1.1 given here is closer to the version provided by Rabin and Scott [28].

In Chapter 2 we consider, briefly, the relative descriptional complexity of converting between two different representations of a regular language. More precisely, we examine the worst-case state complexity of a regular languages relative to the size of its description as an NFA and as a regular expression.

In addition to comparing the complexity of different representations of the same language, we can also consider the state complexity of a language resulting from an operation, relative to the complexity of the operand languages. This topic has been widely studied. Yu and Zhuang [36] and Birget [3] investigated the state complexity of the intersection of regular languages. Their work was then extended by Yu, Zhuang, and Salomaa [35, 37], to include other basic operations such as union, concatenation, reversal, complement, and Kleene closure. Pighizzini and Shallit [24] gave special treatment to union, intersection, and concatenation in the unary case, that is, for an alphabet of size 1.

For regular languages L_1 and L_2 over an arbitrary alphabet Σ, with $\mathrm{sc}(L_1) = m$ and $\mathrm{sc}(L_2) = n$, a summary of the known results is given in Table 1.2. These upper bounds are tight.

Operation	State Complexity
$L_1 \cup L_2$	mn
$L_1 \cap L_2$	mn
$\Sigma^* \backslash L_1$	m
$L_1 L_2$	$m2^n - 2^{n-1}$
L_1^R	2^m
L_1^*	$2^{m-1} + 2^{m-2}$

Table 1.2: State complexity of some basic operations

In this thesis, we focus on the problem of determining the worst-case state complexity of the operation $\mathrm{root}(L)$, given by

$$\mathrm{root}(L) = \{w \in \Sigma^* : \exists n \geq 1 \text{ such that } w^n \in L\},$$

where L is a regular language. This entails a discussion of the monoid of transformations of a finite set. For a language L, over a k-letter alphabet, there is a natural connection between the state complexity of $\mathrm{root}(L)$ and the largest monoid generated by k elements. This problem of the largest monoid has had very little treatment. In fact, until recently, the most interesting case, $k = 2$, seems to have had no treatment at all.

In Chapter 3, we discuss monoids and their various connections to formal languages. We present results concerning the size of the largest monoid generated by a k element set, with a focus on the case for $k = 2$. Much of the work included in this chapter was done independently. However, this author later discovered that these results had recently been published by Holzer and König [9, 10].

In Chapter 4, we show how to construct a language based on a particular monoid, so that we can apply the results of Chapter 3 to obtain bounds (in some cases, tight bounds) on the worst-case state complexity of the operation $\mathrm{root}(L)$. In turn, we use this result to improve the lower bound for the worst-case state complexity of an n-state 2DFA.

Finally, in Chapter 5, we summarize the main results of this thesis, and present some open problems.

Chapter 2

State Complexity of Representations of Regular Languages

As defined in Chapter 1, the set of regular languages is exactly the set of languages that are accepted by a DFA. However, there are other representations to consider.

One such representation is a generalization of a DFA. A *non-deterministic finite automaton*, or *NFA*, is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite non-empty set of states, Σ is the finite input alphabet, $\delta : Q \times \Sigma \to Q^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. For an NFA $\mathcal{A}$, the set $L(\mathcal{A}) = \{w \in \Sigma^* \ : \ \delta(q_0, w) \cap F \neq \emptyset\}$ is said to be the language recognized by $\mathcal{A}$.

Another very different representation of regular languages is based on few basic operations. For languages $X, Y \subseteq \Sigma^*$, define the *concatenation* of X and Y, denoted XY, by the set

$$XY = \{xy \in \Sigma^* : x \in X, y \in Y\}.$$

Define $X^0 = \{\epsilon\}$ and $X^i = XX^{i-1}$ for $i \geq 1$. The *Kleene closure* of a language X, denoted X^*, is given by the set

$$X^* = \bigcup_{i=0}^{\infty} X^i.$$

Then a regular language can be represented by a *regular expression*, defined as follows.

1. $\emptyset$ is a regular expression and denotes the empty set.

2. ϵ is a regular expression and denotes the set $\{\epsilon\}$.

3. For $a \in \Sigma$, a is a regular expression, and denotes the set $\{a\}$.

4. For any regular expressions x and y, that denote the languages X and Y respectively, we have that:

   (a) $(x \cdot y)$, equivalently (xy), is a regular expression and denotes the set XY;

   (b) $(x + y)$ is a regular expression and denotes the set $X \cup Y$;

   (c) (x^*) is a regular expression and denotes the set X^*.

Of the three operations, * has the highest precedence, followed by $\cdot$, and then by $+$. Superfluous parentheses can be omitted.

When considering these alternative forms, a natural question which arises is: what is the efficiency of these representations relative to deterministic finite automata? In other words, when given a representation of a regular language in one of these other forms, what is the worst-case state complexity of the language relative to the complexity of this representation? Of course, in order to consider such a question, we must first decide on a measure of the complexity of these other forms.

An obvious measure of the complexity of a particular NFA is the number of states. This measure is the most common, although other methods could certainly be based on the number of transitions, or any other information unique to a particular NFA.

For a regular expression, since it is a string, the natural choice as a measure seems to be its length. There are, however, a variety of accepted methods for determining the length of a regular expression. One such method, the method we will use, is known as the *alphabetic length*. This is defined as the number of alphabet (Σ) symbols in the expression. Note that the symbol ϵ, denoting the empty string, is not considered to be a symbol of the alphabet, and, therefore, does not contribute to the length of an expression; this choice prevents the artificial padding of an expression to increase its length. Some other methods include the *ordinary length*, the total number of symbols including parenthesis and operators; and the *reverse polish length*, the total number of symbols when the expression is rewritten in parenthesis-free reverse polish notation.

2.1 Deterministic State Complexity of NFAs

The following theorem gives us a tight upper bound on the deterministic state complexity of a language recognized by an NFA.

Theorem 2.1. *For an NFA $\mathcal{A}$, with n states, we have* $\mathrm{sc}(L(\mathcal{A})) \leq 2^n$. *Furthermore, this bound is tight.*

Proof. We have that $|2^Q| = 2^n$, where 2^Q is the set of all subsets of Q. Then the bound $\mathrm{sc}(L(\mathcal{A})) \leq 2^n$ is immediate from the subset construction method for converting an NFA to a DFA given by Rabin and Scott[28], which gives us a DFA with 2^n states recognizing $L(\mathcal{A})$.

To show that this bound is tight, consider the language $L_n = (0 + (01^*)^{n-1}0)^*$. An NFA for L_n is $\mathcal{A} = \{Q, \Sigma, \delta, q_1, F\}$, where $Q = \{q_1, \ldots, q_n\}$, $\Sigma = \{0, 1\}$, $F = \{q_1\}$, and δ is defined as follows:

$$\begin{aligned}
&\delta(q_1, 0) = \{q_1, q_2\};\\
&\delta(q_1, 1) = \emptyset;\\
&\delta(q_i, 0) = \{q_{i+1}\}, \quad \text{for } 2 \leq i < n;\\
&\delta(q_i, 1) = \{q_i\}, \quad \text{for } 2 \leq i \leq n;\\
&\delta(q_n, 0) = \{q_1\}.
\end{aligned}$$

The NFA $\mathcal{A}$ is depicted in Figure 2.2.

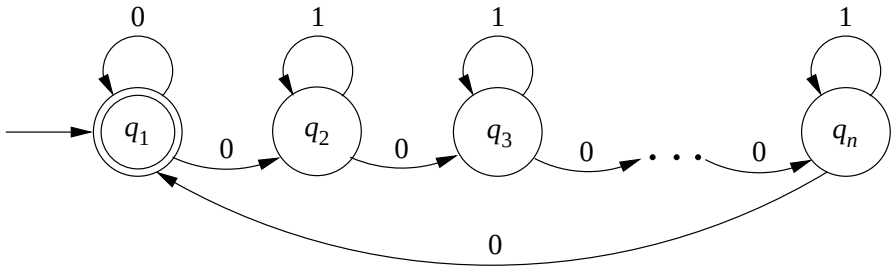

Figure 2.2: The NFA for $(0 + (01^*)^{n-1}0)^*$.

In his study of the ambiguity of nondeterministic finite automata, Leung [17] proved that the language L_n has state complexity 2^n. We give our own version of the proof here.

A set of states $P \subseteq Q$ is called *reachable* if there exists a string u such that $\delta(q_1, u) = P$. Furthermore, for $P' \subseteq Q$, a string x is said to *distinguish* P and P' if one of $\delta(P, x) \cap F \neq \emptyset$ and $\delta(P', x) \cap F \neq \emptyset$ is true, but not both. By $\delta(P, x)$ we mean $\bigcup_{q \in P} \delta(q, x)$. If there is no string that distinguishes P and P', then P and P' are *equivalent*.

In order to show that the smallest DFA recognizing L_n has 2^n states, we must show that each subset of states is reachable, and that no subset of states is equivalent to any other subset of states. These facts imply that there exist 2^n strings that are pairwise inequivalent, with respect to the Myhill–Nerode equivalence relation $\sim_L$ defined at the end of Chapter 1.

For $P \subseteq Q$, define $w_P \in \Sigma^*$ by $w_P = w_1 0 w_n 0 w_{n-1} 0 \cdots 0 w_1$, where $w_i = \epsilon$, if $q_i \in P$, and $w_i = 1$ otherwise. Furthermore, define $u_P \in \Sigma^*$ by $u_P = 0^{n-1} w_P$.

As the following lemma will show, the string w_P gives us a degree of control over the non-determinism in $\mathcal{A}$, by allowing us to cancel out any states not in P. We will then use this fact to show that the string u_P allows us to reach the state P.

Lemma 2.3 (Leung). *For $P \subseteq Q$, we have*

1. $\delta(Q \backslash P, w_P) = \emptyset$;
2. $\delta(P, w_P) = P$.

Proof. To show that $\delta(Q \backslash P, w_P) = \emptyset$, notice that for $q_i \notin P$, if $i > 1$, we have

$$\begin{aligned} \delta(q_i, w_P) &= \delta(q_i, w_1 0 w_n 0 \cdots 0 w_1) \\ &= \delta(q_n, 0 w_i 0 \cdots 0 w_1) \\ &= \delta(q_1, w_i 0 w_{i-1} 0 \cdots 0 w_1) \\ &= \emptyset \quad (\text{since } w_i = 1). \end{aligned}$$

If $i = 1$, then clearly $\delta(q_i, w_P) = \emptyset$, since $\delta(q_1, 1) = \emptyset$. Now, to show that $\delta(P, w_P) = P$, notice that for $q_i \in P$, we have

$$\begin{aligned}
\delta(q_i, w_P) &= \delta(q_i, w_1 0 w_n 0 \cdots 0 w_1) \\
&= \delta(q_n, 0 w_i 0 \cdots 0 w_1) \\
&= \delta(q_1, w_i 0 w_{i-1} 0 \cdots 0 w_1) \\
&= \delta(q_1, 0 w_{i-1} 0 \cdots 0 w_1) \quad (\text{since } w_i = \epsilon) \\
&\supseteq \delta(q_2, w_{i-1} 0 \cdots 0 w_1) \\
&= \delta(q_{1+i-1}, w_1) \\
&= \{q_i\}.
\end{aligned}$$

Then it follows that $\delta(P, w_P) \supseteq P$. Furthermore, for any $q_i \notin P$ if $q_i \in \delta(P, w_P)$, then we must have that $q_1 \in \delta(q_1, w_i 0 \cdots 0 w_1)$. But $w_i = 1$, so $\delta(q_1, w_i 0 \cdots 0 w_1) = \emptyset$. Therefore $\delta(P, w_P) = P$, and the proof is complete. □

It follows from Lemma 2.3 that

$$\delta(q_1, u_P) = \delta(Q, w_P) = \delta(Q \backslash P, w_P) \cup \delta(P, w_P) = P.$$

Hence, every set of states P is reachable. Now, for any $P' \subseteq Q$, let q_i be the largest index i such that q_i is an element of one of P or P', but not both. If $i = 1$, then let $x = \epsilon$, otherwise let $x = (10)^{n-i+1}$. Without loss of generality, assume that $q_i \in P$. Then we have

$$\delta(P, x) \subseteq \delta(q_i, x) = \{q_1\}.$$

However, $\delta(q_j, x) = q_{j+n-i+1} \neq q_1$, for all $1 < j < i$; and $\delta(q_j, x) = \emptyset$, for all $j > i$, or $j = 1$. Then $\delta(P', x) \cup F = \emptyset$, so we can distinguish P and P'. Then by Theorem 1.1, $\mathrm{sc}(L_n) = 2^n$. Hence, the bound of 2^n on the state complexity is tight. □

2.2 Deterministic State Complexity of Regular Expressions

As seen in the previous section, the language L_n can be represented by the regular expression $(0 + (01^*)^{n-1}0)^*$ with alphabetic length $2n$. Then the result of Theorem 2.1 implicitly

gives us a lower bound of $2^{n/2}$ on the worst-case state complexity of a regular expression with alphabetic length n. However, with a small modification to the regular expression, we can use the same proof technique to improve this bound. This fact was shown by Ellul, Krawetz, Shallit, and Wang [8]. A more detailed proof is given here.

Theorem 2.4. *The language* $K_r = (0^*(01^*)^{r-1}0)^*$ *has state complexity* $2^r + 2^{r-2} = \frac{5}{4} \cdot 2^r$.

Proof. An NFA for K_r is $\mathcal{B} = \{Q, \Sigma, \delta, q_0, F\}$, where $Q = \{q_0, q_1, \ldots, q_{r+1}\}$, $\Sigma = \{0, 1\}$, $F = \{q_0, q_{r+1}\}$, and δ is defined as follows:

$$\begin{aligned}
&\delta(q_0, 0) = \delta(q_1, 0) = \{q_1, q_2\};\\
&\delta(q_0, 1) = \delta(q_1, 1) = \delta(q_{r+1}) = \delta(q_{r+1}, 0) = \emptyset;\\
&\delta(q_i, 0) = \{q_{i+1}\}, \quad \text{for } 2 \le i < r;\\
&\delta(q_i, 1) = \{q_i\}, \quad \text{for } 2 \le i \le r;\\
&\delta(q_r, 0) = \{q_0, q_1, q_{r+1}\}.
\end{aligned}$$

The NFA $\mathcal{B}$ is depicted in Figure 2.5.

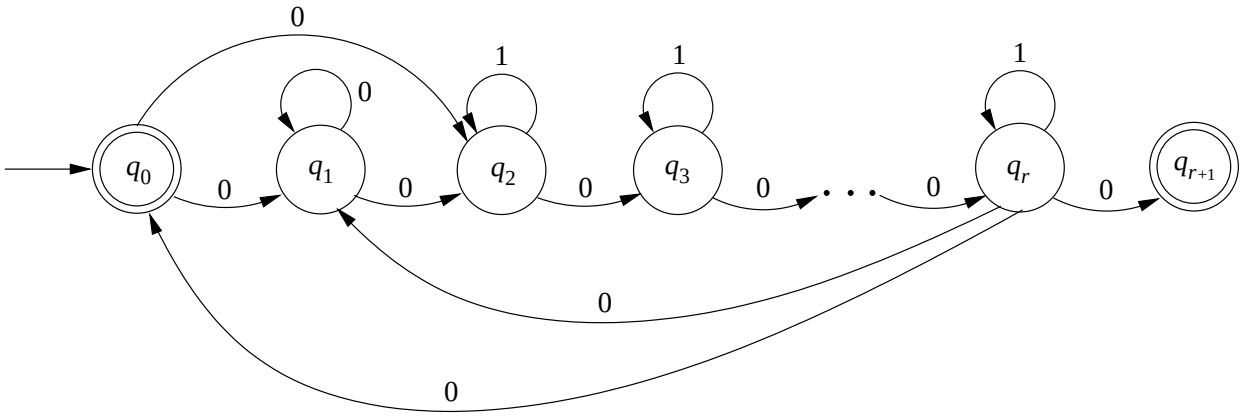

Figure 2.5: The NFA for $(0^*(01^*)^{r-1}0)^*$.

We will show that the only reachable sets of states are those of the form:

(a) $\{q_0\}$;

(b) $\{q_0, q_1, q_{r+1}\} \cup X$, where $X \subseteq \{q_2, q_3, \cdots, q_r\}$;

(c) $\{q_1, q_2\} \cup Y$, where $Y \subseteq \{q_3, q_4, \cdots, q_r\}$;

(d) Z, where $Z \subseteq \{q_2, q_3, \cdots, q_r\}$.

For $P \subset Q$, define $w_P \in \Sigma^*$ by $w_P = 0w_r0w_{r-1}0 \cdots 0w_20$, where $w_i = \epsilon$, if $q_i \in P$, and $w_i = 1$, otherwise.

Lemma 2.6. *For $P = \{q_0, q_1, q_{r+1}\} \cup X$, where $X \subseteq \{q_2, q_3, \cdots, q_r\}$, we have*

1. $\delta(Q \backslash P, w_P) = \emptyset$;
2. $\delta(q_i, 1w_P) \supseteq \{q_i\}$, *when* $q_i \in P$;
3. $\delta(P, w_P) = P$.

Proof. To show that $\delta(Q \backslash P, w_P) = \emptyset$, notice that for $q_i \notin P$, we have

$$\begin{aligned}
\delta(q_i, w_P) &= \delta(q_i, 0w_r0 \cdots 0w_20) \\
&= \delta(q_r, 0w_i0 \cdots 0w_20) \\
&= \delta(\{q_0, q_1, q_{r+1}\}, w_i0w_{i-1}0 \cdots 0w_20) \\
&= \emptyset \quad (\text{since } w_i = 1).
\end{aligned}$$

To show that $\delta(q_i, w_P) \supseteq \{q_i\}$, when $q_i \in P$, notice that for $2 \leq i \leq r$, we have

$$\begin{aligned}
\delta(q_i, w_P) &= \delta(q_i, 0w_n0 \cdots 0w_20) \\
&= \delta(q_r, 0w_i0 \cdots 0w_20) \\
&= \delta(\{q_0, q_1, q_{r+1}\}, w_i0w_{i-1}0 \cdots 0w_20) \\
&\supseteq \delta(q_1, 0w_{i-1}0 \cdots 0w_20) \quad (\text{since } w_i = \epsilon) \\
&\supseteq \delta(q_2, w_{i-1}0 \cdots 0w_20) \\
&= \{q_i\}.
\end{aligned}$$

Furthermore, notice

$$\delta(q_0, w_P) \supseteq \delta(q_r, 0) = \{q_0, q_1, q_{r+1}\}.$$

It follows that $\delta(P, w_P) \supseteq P$. Now, for $2 \leq i \leq r$, if $q_i \in \delta(P, w_P)$, then we must have that $\{q_0, q_1\} \cap \delta(\{q_0, q_1\}, w_i 0 \cdots 0 w_2 0) \neq \emptyset$. This implies that $w_i = \epsilon$, and hence $q_i \in P$. Therefore $\delta(P, w_P) = P$, and the proof is complete. □

Lemma 2.7. *For $P = \{q_1, q_2\} \cup X$, where $X \subseteq \{q_3, q_4, \cdots, q_r\}$, we have*

1. *$\delta(Q \backslash P, 1 w_P) = \emptyset$;*
2. *$\delta(P, 1 w_P) = P$.*

Proof. First, notice that $w_P = w_{P'}$, for some P' of the form required by Lemma 2.6. Then, by Lemma 2.6, for $3 \geq i \leq r$, we have

$$\delta(q_i, 1 w_P) = \delta(q_i, w_P) = \delta(q_i, w_{P'}) = \emptyset, \text{ when } q_i \in P.$$

Clearly $\delta(\{q_0, q_{r+1}\}, 1 w_P) = \emptyset$, and so $\delta(Q \backslash P, 1 w_P) = \emptyset$.

Also by Lemma 2.6, for $3 \leq i \leq r$, we have

$$\delta(q_i, 1 w_P) \supseteq \{q_i\}, \text{ when } q_i \in P.$$

Notice

$$\delta(q_2, 1 w_P) \supseteq \delta(q_r, 0 w_2 0) = \delta(\{q_0, q_1, q_{r+1}, w_2 0) = \{q_1, q_2\}.$$

Then it follows that $\delta(P, 1 w_P) \supseteq P$. Furthermore, for $3 \leq i \leq r$, if $q_i \notin P$ we have $\delta(P, 1 w_P) \subseteq \delta(P', w_{P'}) \not\ni q_i$, by Lemma 2.6. Finally, if $\delta(P, 1 w_P)$ contains q_0 or q_{r+1}, then this implies that $q_r \in \delta(P, 10 w_r 0 \cdots 0 w_2) \Rightarrow q_2 \in \delta(q_0, 1)$, a contradiction. Hence, $\delta(P, 1 w_P 1) = P$, and the proof is complete. □

Lemma 2.8. *For $P \subseteq \{q_2, q_3, \cdots, q_r\}$, we have*

1. *$\delta(Q \backslash P, 1 w_P 1) = \emptyset$;*
2. *$\delta(P, 1 w_P 1) = P$.*

Proof. First, notice that $w_P = w_{P'}$ for some P' of the form required by Lemma 2.6. Then by Lemma 2.6, for $2 \leq i \leq r$, we have

$$\delta(q_i, 1 w_P 1) = \delta(q_i, w_{P'} 1) = \emptyset, \text{ when } q_i \in P.$$

Clearly $\delta(\{q_0, q_1, q_{r+1}\}, 1w_P1) = \emptyset$, and so $\delta(Q\backslash P, 1w_P1) = \emptyset$.

Also by Lemma 2.6, for $2 \leq i \leq r$, we have

$$\delta(q_i, 1w_P1) \supseteq \{q_i\}, \text{ when } q_i \in P.$$

Furthermore, for $2 \leq i \leq r$, if $q_i \notin P$ we have $\delta(P, 1w_P1) \subseteq \delta(P', w_{P'}) \not\ni q_i$, by Lemma 2.6. Finally, $\{q_0, q_1, q_{r+1}\} \cap \delta(P, 1w_P1) = \emptyset$, since $\{q_0, q_1, q_{r+1}\} \cap \delta(Q, 1) = \emptyset$. Hence, $\delta(P, 1w_P1) = P$, and the proof is complete. □

Clearly the set $\{q_0\}$ is reachable. For $P \subseteq \{q_2, q_3, \cdots, q_r\}$, we have $\delta(q_0, 0^r1w_P1) = \delta(Q, 1w_P1) = P$, by Lemma 2.8. Furthermore, for $P = \{q_1, q_2\} \cup X$, with $X \subseteq \{q_3, q_4, \cdots, q_r\}$, we have $\delta(q_0, 0^r1w_P) = \delta(Q, 1w_P) = P$, by Lemma 2.7. And finally, for $P = \{q_0, q_1, q_{r+1}\} \cup X$, with $X \subseteq \{q_2, q_3, \cdots, q_r\}$, we have $\delta(q_0, 0^rw_P) = \delta(Q, w_P) = P$, by Lemma 2.6.

Hence, every set of states P is reachable, when P is in one of the forms (a)–(d) given above. Now, suppose that P is a reachable set of states; that is for some $x\Sigma^*$, we have $\delta(q_0, x) = P$. If $q_0 \notin P$, then to arrive in state q_1 we must have taken the transition $\delta(q_0, 0) = \{q_1, q_2\}$, or $\delta(q_1, 0) = \{q_1, q_2\}$, on the last symbol of x. It follows that $q_2 \in P$. Furthermore, if $q_{r+1} \in P$ then we must have taken the transition $\delta(q_r, 0) = \{q_0, q_1, q_{r+1}\}$ on the last symbol of x. It follows that $q_0, q_1 \in P$. Hence, sets of the form (a)–(d) are the only reachable sets of states. This gives a total of $2^{r-1} + 2^{r-2} + 2^{r-2} + 1$ reachable sets of states.

It is easy to see that the subsets $\{q_0\}$ and $\{q_0, q_1, q_{r+1}\}$ are equivalent in terms of the Myhill-Nerode Theorem. Then for any two sets P, P' of two different forms, we can restrict our discussion to those of the form (b)–(c). Any set of the form (b) can be distinguished from a set of the form (c) or (d) by the string ϵ, since sets of the form (c) or (d) contain no final states. If the set P is of the form (c), and P' is of the form (d), then consider the string $x = 0(10)^r$. Then

$$\delta(P, x) \subseteq \delta(q_1, x) = \{q_0, q_1, q_{r+1}\}.$$

However, $\delta(q_j, x) = \emptyset$, for $1 < j \leq r$, so that $\delta(P', x) = \emptyset$.

For any two different sets P, P' of the same form, let q_i be the largest index i such that q_i is an element of one of P or P', but not both. Then $i \geq 2$. Let $x = (10)^{r-i+1}$. Without

loss of generality, assume that $q_i \in P$. Then we have

$$\delta(P, x) \subseteq \delta(q_i, x) = \{q_0, q_1, q_{r+1}\}.$$

However, $\delta(q_j, x) = q_{j+r-i+1}$, for all $1 < j < i$; and $\delta(q_j, x) = \emptyset$, for all $j > i$ or $j \in \{0, 1\}$. Then $\delta(P', x) \cup F = \emptyset$, so we can distinguish P and P'.

Then by the Myhill-Nerode theorem, the state complexity of the language K_r is $2^r + 2^{r-2}$. □

Since the language K_r is represented by the regular expression $(0^*(01^*)^{r-1}0)^*$, which has alphabetic length $2r = n$, this gives us a worst-case lower bound of $\frac{5}{4} \cdot 2^{n/2}$ on the state complexity of a regular expression of alphabetic length n. Currently, the best known upper bound is $2^n + 1$, and so the problem of finding a tight upper bound remains open (see Open Problem 5.1).

Chapter 3

Monoids

We leave the topic of state complexity for the moment to consider the monoid, an object from abstract algebra with significant connections to formal languages. Much like a group, a monoid is a set of elements with an operation for composing those elements. However, unlike a group, it is not required that every element of the the monoid have an inverse.

Formally, a *monoid* $(M, \cdot)$ is a set M with a binary operation, denoted by $\cdot$. The operation $\cdot$ is associative, and the set M is closed under this operation. Furthermore, M contains a unique *identity* element, denoted by 1, such that $1 \cdot x = x \cdot 1 = x$, for all $x \in M$. For a set $X \subseteq M$, if X is closed under the operations $\cdot$, then $(X, \cdot)$ is a *submonoid* of $(M, \cdot)$.

If for some $X \subseteq M$, we have that M is the closure of X under the operation $\cdot$, then we say that the set X *generates* the monoid $(M, \cdot)$ and we write $\text{span}(X) = M$. The elements of X are known as *generators*.

When the context is clear, and there is no uncertainty regarding the implied binary operation, we often denote the monoid $(M, \cdot)$ simply by M.

For $x \in M$, if there exists $y \in M$ such that $x \cdot y = 1$, we say that y is the *inverse* of x, and we write $y = x^{-1}$. If such an element exists, it is unique. Furthermore, we have that x is the inverse of y.

A simple example of a monoid is the set of non-negative integers $\mathbb{N}$ with the addition operation, denoted $+$. It is clear that $+$ is associative and that the set $\mathbb{N}$ is closed under this operation. Here, the element 0 is the unique identity element.

The most common example of a monoid involves the functions that map a set into itself.

For a set Q, a function $f : Q \to Q$ is called a *transformation*. The set of all transformations of Q is denoted by Q^Q. For transformations $f, g \in Q^Q$, their composition is written $f \cdot g$, and is given by $(f \cdot g)(q) = g(f(q))$, for all $q \in Q$. With composition defined in this way, it is clear that the transformation $i : Q \to Q$, given by $i(q) = q$ for all $q \in Q$, serves as the identity element. Then the set Q^Q and the composition operator form a monoid.

Transformations and their monoids have been studied in some detail by Dénes (whose work is summarized in [6]), and Salomaa [31, 32]. Dénes investigated several algebraic and combinatorial properties of transformations, while much of Salomaa's work is concerned with subsets that generate the full monoid of transformations.

It is the monoid that provides the major link between formal languages and algebraic theory. In fact, for a finite non-empty alphabet Σ, the set Σ^* is itself a monoid. That is, the set of strings Σ^* together with the concatenation operation form a monoid, with the empty string ϵ as the identity element. This monoid is known as the *free monoid over the alphabet* Σ.

A more interesting occurrence of a monoid in formal languages describes the behaviour of a DFA. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA. For $w \in \Sigma^*$, define the function $\delta_w : Q \to Q$ by $\delta_w(q) = \delta(q, w)$, for all $q \in Q$. If we denote the empty word by ϵ, then δ_ϵ is the identity function. Define the composition operation, denoted $\cdot$, by $(f \cdot g)(q) = g(f(q))$, for all $q \in Q$. Then the set of all functions δ_w together with the composition operator form a monoid, where δ_ϵ is the identity element. This is called the *transition monoid* of $\mathcal{A}$.

It is not hard to see that for any $x, y \in \Sigma^*$ we have $\delta_{xy} = \delta_x \cdot \delta_y$. So the transition monoid of $\mathcal{A}$ is generated by the set $\{\delta_a \in Q^Q : a \in \Sigma\}$.

Example 3.1. For the DFA $\mathcal{M}$, depicted in Figure 3.2, we have

$$\delta_a(q) = \begin{cases} 1 & \text{if } q = 1, \\ 3 & \text{if } q \in \{2, 3\}; \end{cases} \qquad \text{and} \qquad \delta_b(q) = \begin{cases} 2 & \text{if } q = 1, \\ 1 & \text{if } q \in \{2, 3\}. \end{cases}$$

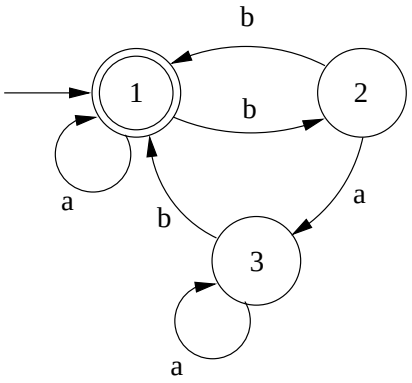

Figure 3.2: The automaton $\mathcal{M}$.

Then $(X, \cdot)$ is the transition monoid of $\mathcal{M}$, where $X = \{\delta_\epsilon, \delta_a, \delta_b, \delta_{ba}, \delta_{bb}\}$.

In Chapter 4, we will see an application of the results discussed in this chapter to the transition monoid of a regular language. This will give us new bounds concerning the state complexity of an operation on regular languages.

Possibly the most important example of a monoid in formal languages is based on a congruence relation of a language. Let L be a language over the alphabet Σ. The *syntactic congruence* of L is the relation $\approx_L$, where $x \approx_L y$ for $x, y \in \Sigma^*$ if and only if for all $u, v \in \Sigma^*$ we have $uxv \in L \Leftrightarrow uyv \in L$. For $x \in \Sigma^*$, let $[x]_L$ denote the equivalence class of $x \in \Sigma^*$. Then for $y \in \Sigma^*$ let the concatenation of the equivalence classes be given by $[x]_L[y]_L = [xy]_L$. Then the quotient $\Sigma^*/\approx_L$, with concatenation, forms the *syntactic monoid of* L. It is well known [14, Chap. 6, Prop. 1.8] that if $\mathcal{A}$ is the minimal DFA recognizing L, then the syntactic monoid of L is isomorphic to the transition monoid of $\mathcal{A}$.

The study of the syntactic monoid forms the basis of the the so-called "algebraic theory" of formal languages. A good overview can be found in [26].

In this chapter, we investigate the problem of determining the largest monoid generated by a finite set of elements. However, before we can begin that discussion, we must first define some notation and terminology.

3.1 The Monoid of Transformations of a Finite Set

Define $Z_n = \{1, 2, \ldots, n\}$, and let $T_n = Z_n^{Z_n}$ be the set of transformations of Z_n. Note that $|T_n| = n^n$.

For $\gamma \in T_n$ we write

$$\gamma = \begin{pmatrix} 1 & 2 & \cdots & n \\ \gamma(1) & \gamma(2) & \cdots & \gamma(n) \end{pmatrix}.$$

When composing elements of T_n, we often omit the composition operator. Instead, composition is implied by juxtaposition. That is, for $\gamma, \delta \in T_n$ we have $\gamma \cdot \delta = \gamma\delta$. When an element is composed with itself k times, where k is a positive integer, we write γ^k; *e.g.*, $\gamma^3 = \gamma\gamma\gamma$. If γ has an inverse δ, we write $\delta = \gamma^{-1}$. Then for a positive integer k, we have $\gamma^{-k} = (\gamma^{-1})^k$. The element γ^0 denotes the identity transformation.

Definition. The *order* of a transformation γ is the size of the set generated by $\{\gamma\}$, and is denoted by $\mathrm{ord}(\gamma)$.

Definition. For $\gamma \in T_n$, define the *image* of γ by

$$\mathrm{img}(\gamma) = \{y \in Z_n : y = \gamma(z) \text{ for some } z \in Z_n\}.$$

Definition. For $\gamma \in T_n$, define the *rank* of γ as the number of distinct elements in the image of γ, and denote it by $\mathrm{rank}(\gamma)$. That is, $\mathrm{rank}(\gamma) = |\mathrm{img}(\gamma)|$.

Example 3.3. Let $\gamma \in T_5$ be given by

$$\gamma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 4 & 5 & 3 \end{pmatrix}.$$

Then $\mathrm{img}(\gamma) = \{1, 3, 4, 5\}$ and $\mathrm{rank}(\gamma) = 4$. Furthermore,

$$\{\gamma^0, \gamma^1, \gamma^2, \ldots\} = \{\gamma^0, \gamma^1, \gamma^2, \gamma^3\},$$

so $\mathrm{ord}(\gamma) = 4$.

When considering the transformations of a finite set, we can safely restrict the discussion to transformations of the set Z_n, since every finite monoid is isomorphic to a submonoid of T_n for some n. In fact, every monoid is isomorphic to a transformation monoid [25, Chap. 1, Prop. 1.5].

3.2 The Group of Permutations of a Finite Set

A *group* $(G, \cdot)$ is a monoid where every element has an inverse. If $H \subseteq G$ is also a group, then H is a *subgroup* of G.

Let $\gamma \in T_n$ be a bijection; in other words, suppose $\text{rank}(\gamma) = n$. Then γ is called a *permutation*. Define $S_n \subseteq T_n$ to be the set of all permutations of Z_n. Then $(S_n, \cdot)$ is a submonoid of $(T_n, \cdot)$. In fact, S_n is a group, and is known as the *symmetric group*. Note that $|S_n| = n!$.

Let $X = \{x_1, \ldots, x_k\}$, where $k < n$, be a subset of Z_n. Let $\gamma \in S_n$ be given by

$$\gamma(x) = \begin{cases} x_{(i+1) \bmod k} & \text{if } x = x_i \in X, \\ x \text{ otherwise.} \end{cases}$$

Then γ is called a *cycle*, in particular, a *k-cycle*. For γ, we write $\gamma = (x_1\ x_2\ \cdots\ x_k)$. Appropriately, this notation is called *cycle notation*. When γ is a 2-cycle, it is known as a *transposition*.

In general, any permutation can be uniquely expressed as a composition of disjoint cycles. Suppose that $\gamma = \gamma_1\gamma_2\cdots\gamma_m$, is such a decomposition. Then γ can be expressed in cycle notation as the juxtaposition of the cycle notation of each γ_i.

Example 3.4. Let $\gamma \in S_7$ be given by $\gamma = \gamma_1\gamma_2\gamma_3$, where $\gamma_1 = (1\ 3\ 2)$, $\gamma_2 = (4\ 5)$, and $\gamma_3 = (6\ 7)$, then in cycle notation, $\gamma = (1\ 3\ 2)(4\ 5)(6\ 7)$.

For a more in-depth discussion of permutations and permutation groups, consult an abstract algebra text such as [1].

We will now mention, briefly, a few results concerning the size of the largest groups generated by a finite set. Since it is well known that the group S_n can be generated by the set $X = \{a, b\}$, where a is any n-cycle and b is any transposition, the problem is solved for sets of size two or more. We can still ask, however, for a characterization of these generating sets. One such result was proved by Piccard [23, Part 1, Prop. 42].

Theorem 3.5 (Piccard). *Except when $n = 4$ and a is one of $(1\ 2)(3\ 4)$, $(1\ 3)(2\ 4)$, or $(1\ 4)(2\ 3)$, if $a \in S_n$ is not the identity permutation, then there exists $b \in S_n$ such that the set $\{a, b\}$ generates S_n.*

In the case of the largest group generated by a single element, consider the function $g(n)$ given by

$$g(n) = \max_{\gamma \in S_n} \{\mathrm{ord}(\gamma)\}. \qquad (*)$$

This function is known as Landau's function, and has been well studied (for a survey, see [19, 22]). Probably the earliest significant result concerning the function g, was obtained by Landau [15], who proved that

$$\log g(n) \sim \sqrt{n \log n}.$$

3.3 Monoids Generated by Finite Sets

In this section, we examine the problem of determining the size of the largest monoid generated by a set of size m. We begin with the following well known result:

Lemma 3.6. *Let $n \geq 3$. Suppose $H \subseteq T_n$ and H generates T_n. Then $|H| \geq 3$. Furthermore, $|H| = 3$ if and only if H can be written in the form $H = \{a, b, c\}$, where $\{a, b\}$ generates S_n and $\mathrm{rank}(\gamma) = n - 1$.*

For a proof of this lemma, see Dénes [5]. This gives us the result that the largest submonoid generated by three elements has the full size n^n, and thus solves the problem for $m \geq 3$. Furthermore, as demonstrated in the next lemma, there is a gap of at least $\binom{n}{2}$ between the size of T_n, and the largest proper submonoid of T_n. This fact is independent of the number of generators. Though this gap is almost certainly not tight, it is large enough for its intended application in the following chapter.

Lemma 3.7. *For $n \geq 1$, if $M \subseteq T_n$ is a monoid such that $|M| > n^n - \binom{n}{2}$, then $M = T_n$.*

Proof. For $1 \leq n \leq 3$, the result can easily be verified computationally, so assume that $n \geq 4$.

Since $|M| > |T_n| - \binom{n}{2}$, there are at most $\binom{n}{2} - 1$ elements of T_n missing from M. There are $\binom{n}{2}$ transpositions in T_n. Then it follows that M must contain at least one transposition. Also, there are $(n-1)!$ permutations of Z_n whose cycle structure consists of

a single cycle with length n. Since $n \geq 4$, we have that $(n-1)! \geq \binom{n}{2}$. Again, considering the size of M, it follows that M must contain at least one n-cycle. Hence $S_n \subseteq M$.

Now consider the transformations in T_n with rank $n-1$. For a transformation to have rank $n-1$, two elements of Z_n must have the same image; this can be done in $\binom{n}{2}$ ways. Furthermore, we must have that one element of Z_n is missing from the image; this can be done in n ways. Finally, the $n-1$ elements of the image can be arranged in $(n-1)!$ ways. This gives a total of $\binom{n}{2} \cdot n!$. It follows that M must contain at least one transformation of rank $n-1$. Then by Lemma 3.6, we have that $M = T_n$. □

The case for $m = 1$ has been studied in connection with Landau's function. Szalay [34] showed that the largest submonoid of T_n generated by a single element has size

$$\exp\left\{\sqrt{n\left(\log n + \log\log n - 1 + \frac{\log\log n - 2 + o(1)}{\log n}\right)}\right\}.$$

In the case $m = 2$, the determining of the largest monoid on m generators is considerably more involved. This case is the focus for the remainder of this chapter.

3.3.1 Sets of Size Two: The Single Cycle Case

Before we examine the full problem of determining the largest monoid on two generators, we focus on the case where one of the generators is a cycle, and the other is a non-bijective transformation. The study of this particular class of monoids gives us some insight into how the properties of the generators of a monoid interact to determine its elements. This class, however, provides more than just an interesting example of two-generated monoids – it also has a role in the proof of the largest monoid.

Let $n \geq 4$, and let $1 \leq p < \frac{n}{2}$ be an integer such that $p \mid n$. Now let $a, b \in T_n$, where

$$a = \begin{pmatrix} 1 & 2 & 3 & \cdots & n-3 & n-2 & n-1 & n \\ 2 & 3 & 4 & \cdots & n-2 & n-1 & n & 1 \end{pmatrix},$$

and if p odd, define

$$b = \begin{pmatrix} 1 & 2 & \cdots & p & p+1 & p+2 & \cdots & n-2 & n-1 & n \\ 1 & 2 & \cdots & p & 1 & p+1 & \cdots & n-3 & n-1 & n-2 \end{pmatrix},$$

while if p even, define

$$b = \begin{pmatrix} 1 & 2 & \cdots & p & p{+}1 & p{+}2 & \cdots & n{-}2 & n{-}1 & n \\ 1 & 2 & \cdots & p & 1 & p{+}1 & \cdots & n{-}3 & n{-}2 & n{-}1 \end{pmatrix}.$$

When $n = 4$ and $p = 2$, or when $n = 6$ and $p = 3$, the above definition of b is somewhat ambiguous. In these two cases, respectively, we take

$$b = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 1 & 3 \end{pmatrix}, \text{ and } b = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 1 & 5 & 4 \end{pmatrix}.$$

Let V_n^p denote the monoid generated by a and b. Furthermore, define $\alpha = ab$, $\beta = a^{p+1}b$, and $\gamma = b$, all restricted to the domain Z_{n-1}.

Lemma 3.8. *We have* $\operatorname{span}(\alpha, \beta, \gamma) = T_{n-1}$.

Proof. We begin by proving that $S_{n-1} \subseteq \operatorname{span}(\alpha, \beta)$. Note that it suffices to show that $\operatorname{span}(\alpha, \beta)$ contains an $(n{-}1)$-cycle and a transposition.

Define $q = \frac{n}{p}$. Then $2 \leq q \leq n$.

For odd p, we have

$$\alpha = \begin{pmatrix} 1 & 2 & \cdots & p{-}1 & p & p{+}1 & \cdots & n{-}3 & n{-}2 & n{-}1 \\ 2 & 3 & \cdots & p & 1 & p{+}1 & \cdots & n{-}3 & n{-}1 & n{-}2 \end{pmatrix},$$

In cycle notation we can write this as $\alpha = (1\ 2\ 3\ \cdots\ p)(n-2\ n-1)$. Since p is odd, we have $\alpha^p = (n-2\ n-1)$, a transposition. For β, first notice that

$$a^{p+1} = \begin{pmatrix} 1 & 2 & \cdots & n{-}p{-}1 & n{-}p & n{-}p{+}1 & \cdots & n{-}1 & n \\ p{+}2 & p{+}3 & \cdots & n & 1 & 2 & \cdots & p & 1 \end{pmatrix},$$

which gives

$$\beta = \begin{pmatrix} 1 & 2 & \cdots & n{-}p{-}3 & n{-}p{-}2 & n{-}p{-}1 & n{-}p & n{-}p{+}1 & \cdots & n{-}1 \\ p{+}1 & p{+}2 & \cdots & n{-}3 & n{-}1 & n{-}2 & 1 & 2 & \cdots & p \end{pmatrix}.$$

If $p = 1$, then in cycle notation we can write $\beta = (1\ 2\ \cdots\ n{-}3\ n{-}1)$. Then

$$\beta\alpha^p = (1\ 2\ \cdots\ n{-}3\ \ n{-}2\ n{-}1),$$

an $(n-1)$-cycle.

Now, assume $p > 1$. Notice that $\beta(i) = i + p$ for $1 \leq i \leq n - 2p$. Then $q - 2$ applications of β will send the element i successively to $i + p, i + 2p, \ldots, i + (q - 3)p$, and finally $i+(q-2)p$. If $i = p-1$, then $i+(q-2)p = n-p-1$. Then since $\beta(n-p-1) = n-2$ and $\beta(n - 2) = p - 1$, it follows that β contains the q-cycle

$$\beta_q = (p-1 \;\; 2p-1 \;\; \cdots \;\; n-p-1 \;\; n-2).$$

Similarly, if $1 \leq i \leq p-3$ then $\beta(i+(q-2)p) = i+(q-1)p = n-p+i$ and $\beta(n-p+i) = i+1$. Furthermore, if $i = p - 2$ then $\beta(i + (q - 2)p) = n - 1$ and $\beta(n - 1) = p$. And finally, if $i = p$ then $\beta(i + (q - 2)p) = 1$. Then it follows that β contains the $(n-q-1)$-cycle

$$\beta_{n-q-1} = (1 \;\; p+1 \;\; \cdots \;\; n-p+1 \;\; 2 \;\; \cdots \;\; n-p-2 \;\; n-1 \;\; p \;\; \cdots \;\; n-p).$$

Hence $\beta = \beta_{n-q-1}\beta_q$.

Recall that $\alpha^p = (n - 2 \; n - 1)$. Now $n - 2$ is in β_q, and $n - 1$ is in β_{n-q-1}. Then it follows that

$$\beta\alpha^p = (1 \;\; \cdots \;\; n-p-2 \;\; n-2 \;\; p-1 \;\; \cdots \;\; n-p-1 \;\; n-1 \;\; p \;\; \cdots \;\; n-p),$$

an $(n-1)$-cycle.

Then when p is odd and $1 \leq p \leq \frac{n}{2}$, we have that α^p is a transposition and $\beta\alpha^p$ is an $(n-1)$-cycle. Hence $\text{span}(\alpha^p, \beta\alpha^p) = S_{n-1}$.

For even p, we have that

$$\alpha = \begin{pmatrix} 1 & 2 & \cdots & p-1 & p & p+1 & \cdots & n-3 & n-2 & n-1 \\ 2 & 3 & \cdots & p & 1 & p+1 & \cdots & n-3 & n-2 & n-1 \end{pmatrix},$$

In cycle notation we can write this as $\alpha = (1 \; 2 \; \cdots \; p)$. For β we have

$$\beta = \begin{pmatrix} 1 & 2 & \cdots & n-p-3 & n-p-2 & n-p-1 & n-p & n-p+1 & \cdots & n-1 \\ p+1 & p+2 & \cdots & n-3 & n-2 & n-1 & 1 & 2 & \cdots & p \end{pmatrix},$$

so that $\beta(i) = (p + i) \mod n - 1$. It follows that β is an $(n-1)$-cycle. Then

$$\beta^{n-q-1} = (1 \;\; n-1 \;\; n-2 \;\; \cdots \;\; 2) \quad \text{and} \quad \beta^q = (1 \;\; 2 \;\; 3 \;\; \cdots \;\; n-1).$$

This gives

$$\beta^{n-q-1}\cdot\alpha\cdot\beta^q = (2\ \ 3\ \cdots\ p{+}1) \Rightarrow (\beta^{n-q-1}\cdot\alpha\cdot\beta^q)^{-1} = (2\ \ p{+}1\ \cdots\ 3),$$
$$\Rightarrow \alpha^2(\beta^{n-q-1}\cdot\alpha\cdot\beta^q)^{-1} = (1\ \ 2\ \cdots\ p{-}1)(p\ \ p{+}1).$$

But p is even, so $p-1$ is odd. This gives $(\alpha^2(\beta^{n-q-1}\cdot\alpha\cdot\beta^q)^{-1})^{p-1} = (p\ \ p{+}1)$, a transposition. Hence $\text{span}(\alpha,\beta) = S_{n-1}$.

In either case, we have that $\text{span}(\alpha,\beta) = S_{n-1}$. It follows from Lemma 3.6 that $\text{span}(\alpha,\beta,\gamma) = T_{n-1}$. □

The following lemma gives a characterization of the elements in V_n^p.

Lemma 3.9. *We have*

$$V_n^p = \{c \in T_n : c(j) = c(\overline{j+p}) \text{ for some } j \in Z_n\} \cup \{a^i : 1 \le i \le n\},$$

where $\overline{j+p} = (j+p) \mod n$.

Proof. Let $c \in T_n$. If $\text{rank}(c) = n$, then since $\text{rank}(b) < n$, we have that $c \in V_n^p$ if and only if $c = a^i$ for some i. So assume that that $c \in T_n \backslash S_n$.

If $c \in V_n^p$, then since $c \notin S_n$ we must have that $c = a^i b c'$ for some $i \in Z_n$ and some $c' \in V_n^p$. Furthermore, for $j = n - i + 1$, we have $a^i(j) = 1$ and $a^i(\overline{j+p}) = 1 + p$. Then $a^i b(j) = 1$ and $a^i b(\overline{j+p}) = 1$. It follows that $a^i b c'(j) = a^i b c'(\overline{j+p})$.

Now suppose $c \in T_n \backslash S_n$ with $c(j) = c(\overline{j+p}) = r$ for some j. Then there exists $x \in Z_n$ such that $a^x(j) = 1$ and $a^x(\overline{j+p}) = 1 + p$. Furthermore, since $\text{rank}(c) < n$ there exists some $s \in Z_n \backslash \text{img}(c)$. Let $y \in Z_n$ be such that $a^y(s) = n$.

Notice that $\text{img}(ca^y) \subseteq Z_{n-1} = \text{img}(a^x b)$. Then by Lemma 3.8, there exists $d \in V_n^p$ such that $a^x b d = ca^y$, and hence $a^x b d a^{-y} = ca^y a^{-y} = c$. It follows that $c \in V_n^p$. □

An important fact regarding this family of V_n^p monoids is that, as shown in Lemma 3.8, if its elements are restricted to the domain and range Z_{n-1}, then these restricted elements can generate the monoid T_{n-1}. This fact makes it easy for us to characterize which elements are contained in V_n^p. The problem is reduced to considering how the permutation a can arrange the elements of Z_n before the non-bijective transformation b is first applied, and how a can arrange the elements of Z_n after b is last applied. It was an attempt to exploit

this fact that led to the discovery of the monoid $U_{k,l}$ described later in this chapter. Let us now turn the discussion toward the analysis of the size of these V_n^p monoids.

For any function $f : Z_n \to Z_n$ in T_n, we can view the elements in the domain Z_n as vertices of a graph, and we can view the elements in the range Z_n as colours. This gives a natural bijection between transformations and vertex colourings of a graph.

A *proper* colouring of a graph is a vertex colouring such that no two adjacent vertices have the same colour. Define the function $f(n, q)$ to be the number of proper colourings of a q-gon, using n colours. A well known result from graph theory [29, Thm. 6] gives us

$$f(n, q) = (n - 1)^q + (-1)^q(n - 1).$$

Note that the formula for $f(n, q)$ holds even when $q = 2$, if we take a 2-gon to be a graph with two vertices and a single edge. This gives us the following fact concerning the size of the monoid V_n^p.

Theorem 3.10. *For $n \geq 4$, and $1 \leq p < \frac{n}{2}$. If $p \mid n$, and $q = \frac{n}{p}$, then*

$$|V_n^p| = s(n, q) = n^n - (f(n, q))^p + n.$$

Proof. Construct a graph G in the following way: take the elements of Z_n to be the vertices, and for every vertex $x \in Z_n$, add the edge $(x, \overline{x - p})$, and the edge $(x, \overline{x + p})$. Since $pq = n$, the resulting graph G will consist of p components. Each component of the graph will be a q-gon, with the elements of $\{i, i+p, \ldots, i+(q-1)p\}$ as its vertices for some $i \in \{1, \ldots, q\}$. Figure 3.11 shows this construction; note that the dotted edges indicate the edges and vertices that are not explicitly shown.

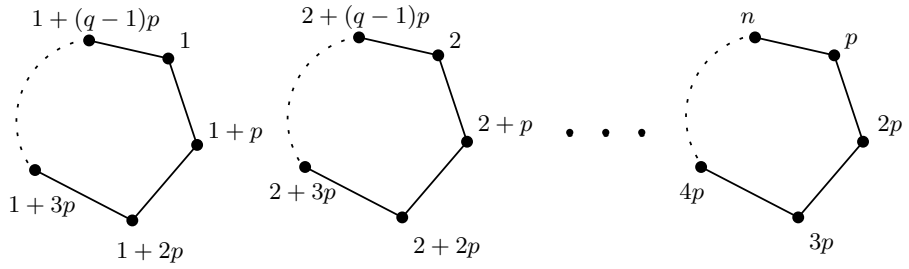

Figure 3.11: The graph G.

Let $C = \{c \in T_n : c(j) = c(\overline{j+p}) \text{ for some } j \in Z_n\}$. From the graph G, it is clear that for $c \in T_n$, we have $c \in C$ if and only if c is not a proper colouring of G. However, since the components of G can be properly coloured independently (each one in $f(n,q)$ ways) we have that G can be properly coloured in $f(n,q)^p$ ways. It follows that $C = n^n - f(n,q)^p$. Now since a has order n, we have, by Lemma 3.9,

$$|V_n^p| = s(n,q) = n^n - (f(n,q))^p + n.$$

□

Now that we have a formula for the size of these V_n^p monoids, we can determine what value of p produces the largest monoid.

Lemma 3.12. *Let $n = 2^i$. Then*

$$s(n,n) > s(n,j) \quad \textit{for all } j \mid n,\ j > 1.$$

Proof. It suffices to show that

$$f(n,n) < f(n,j)^{\frac{n}{j}} \quad \text{for all } j \mid n,\ j > 1.$$

For $n = 2^i$ and $j \mid n$, $j > 1$, we have that n and j are even. Then

$$\begin{aligned} f(n,j)^{\frac{n}{j}} &= ((n-1)^j + n - 1)^{\frac{n}{j}} \\ &> ((n-1)^j)^{\frac{n}{j}} + n - 1 \\ &= (n-1)^n + n - 1 \\ &= f(n,n), \end{aligned}$$

as required. □

Lemma 3.13. *If n is not a power of 2, then let q be the least prime such that $q > 2$ and $q \mid n$. Then*

$$s(n,q) \geq s(n,j) \quad \textit{for all } j \mid n,\ j > 1.$$

Proof. It suffices to show that

$$f(n,q)^{\frac{n}{q}} < f(n,j)^{\frac{n}{j}} \quad \text{for all } j \mid n,\ j > 1.$$

Since q is odd, we have

$$\begin{aligned} f(n,q)^{\frac{n}{q}} &= ((n-1)^q - n + 1)^{\frac{n}{q}} \\ &< ((n-1)^q)^{\frac{n}{q}} \\ &= (n-1)^n. \end{aligned}$$

If j is even, then $f(n,j)^{\frac{n}{j}} > (n-1)^n$. Assume that j is odd. Then

$$f(n,j) = (n-1)^j - (n-1).$$

Furthermore

$$\begin{aligned} f(n,q)^{\frac{j}{q}} &= ((n-1)^q - (n-1))^{\frac{j}{q}} \\ &< (n-1)^j - (n-1)^{\frac{j}{q}}. \end{aligned}$$

Since $j > q$, it follows that

$$f(n,q)^{\frac{j}{q}} < (n-1)^j - (n-1) = f(n,j),$$

and hence

$$f(n,q)^{\frac{n}{q}} < f(n,j)^{\frac{n}{j}}.$$

□

For a fixed value of n, define V_n to be largest monoid of the form V_n^p.

Theorem 3.14. *For $n \geq 4$, if $n = 2^i$ then let $q = n$, otherwise let $q = \min\{q \mid n : q \text{ is odd}\}$. Now let $p = \frac{n}{q}$. Then $V_n^p = V_n$.*

Proof. The result follows immediately from a combination of Lemma 3.12, and Lemma 3.13. □

This work was completed in April 2003 by Krawetz, Lawrence, and Shallit. However, in June 2003 this author discovered that similar results were obtained by Holzer and König [9, 10]. Actually, the work of Holzer and König contains a few more results concerning the size of the monoid V_n, both relative and absolute. A summary is provided here.

Lemma 3.15 (Holzer and König). *For the family of monoids of the form V_n, we have*

$$\lim_{n\to\infty} \frac{|V_n|}{n^n} = 1 - \frac{1}{e}.$$

Lemma 3.16 (Holzer and König). *Let $a \in S_n$ be a cycle of any length, and let $b \in T_n \backslash S_n$. If X is the monoid generated by $\{a, b\}$, then $|X| \leq |V_n|$.*

Lemma 3.17 (Holzer and König). *Let $a, b \in S_n$. If X is the monoid generated by $\{a, b\}$, then $|X| \leq |V_n|$.*

Lemma 3.18 (Holzer and König). *Let $a, b \in T_n \backslash S_n$. If X is the monoid generated by $\{a, b\}$, then $|X| \leq |V_n|$.*

Lemma 3.17 and Lemma 3.18 show us that the largest monoid on two generators must have one permutation and one non-bijective function as its generators. This gives us the following trivial upper bound. Here, $g(n)$ is Landau's function, $(*)$ in Section 3.2.

Corollary 3.19 (Holzer and König). *Let $U \in T_n$ be the largest monoid generated by two generators. Then*

$$|U| \leq n^n - n! + g(n).$$

Lemmas 3.16-3.18 prove that the monoid V_n, is actually the largest possible in a number of different cases. This reduces the search for the overall largest monoid to the case where a is an arbitrary permutation and b is a non-bijective transformation.

3.3.2 Sets of Size Two: The General Case

If a is an arbitrary permutation and b a non-bijective transformation, then we have two possibilities: either b identifies two elements from the same cycle in a, or b identifies two elements from different cycles in a. Let us continue by considering the case where a is

a permutation with exactly two cycles, and b is a transformation that identifies only two elements, one from each cycle in a. The intuition behind the following choice for the generators a and b is that we want to maximize the number of elements generated by a alone, but we still want to be able to generate the full monoid of transformations T_{n-1}, when the elements of $\text{span}(a, b)$ are restricted to Z_{n-1}.

Let $n \geq 5$. Define $k, l \in Z_n$ such that $\gcd(k, l) = 1$, $k + l = n$, and $l > k > 1$. Furthermore, define $a, b \in T_n$ by

$$a = \begin{pmatrix} 1 & 2 & \cdots & k{-}1 & k & k{+}1 & \cdots & n{-}1 & n \\ 2 & 3 & \cdots & k & 1 & k{+}2 & \cdots & n & k{+}1 \end{pmatrix},$$

and when $k = 2$, or l is even

$$b = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & n{-}1 & n \\ k{+}1 & 2 & 3 & 4 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

otherwise

$$b = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & n{-}1 & n \\ k{+}1 & 3 & 2 & 4 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

In cycle notation, we have $a = (1 \ \cdots \ k)(k{+}1 \ \cdots \ n)$. Let $U_{k,l}$ denote the monoid generated by a and b.

Now define x, $1 \leq x \leq k$ such that $lx \equiv -1 \pmod{k}$. Furthermore, define $\alpha = ab$, $\beta = a^{x+1}b$, and $\gamma = b$ all restricted to the domain Z_{n-1}.

Lemma 3.20. *For $n \geq 5$, we have* $\text{span}(\alpha, \beta, \gamma) = T_{n-1}$.

Proof. First, we will show that $S_{n-1} \subseteq \text{span}(\alpha, \beta)$. To prove this fact, recall that it suffices to show that $\text{span}(\alpha, \beta)$ contains a cycle of length $n - 1$ and a transposition.

If $k \neq 2$, and l odd, it is easy to see that

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & \cdots & k{-}1 & k & k{+}1 & \cdots & n{-}2 & n{-}1 \\ 3 & 2 & 4 & \cdots & k & k{+}1 & k{+}2 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

Note that $\alpha \in S_{n-1}$. In cycle notation, we have

$$\alpha = (1\ 3\ 4\ \cdots\ n{-}1).$$

So α is in fact a cycle of length $n-2$.

For β, notice that

$$\beta(i) = \begin{cases} b(i) & 1 \leq i \leq k, \\ ab(i) & k+1 \leq i \leq n-1, \end{cases}$$

giving us

$$\beta = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & k & k{+}1 & \cdots & n{-}2 & n{-}1 \\ k{+}1 & 3 & 2 & 4 & \cdots & k & k{+}2 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

Note that $\beta \in S_{n-1}$. In cycle notation, we have

$$\beta = (1\ \ k{+}1\ \ k{+}2\ \ \cdots\ \ n{-}1)(2\ \ 3).$$

So β has a cycle of odd length l and a cycle of length 2. It follows that, $\beta^l = (2\ 3)$, a transposition. Furthermore, since $\alpha\beta^l = (1\ 2\ 3\ 4\ \cdots\ n{-}1)$ is a cycle of length $n-1$, we have $\operatorname{span}(\alpha\beta^l, \beta^l) = S_{n-1}$. Hence $\operatorname{span}(\alpha, \beta) = S_{n-1}$.

If $k = 2$, or l even, it is easy to see that

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & \cdots & n{-}2 & n{-}1 \\ 2 & 3 & 4 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

Note that $\alpha \in S_{n-1}$. In cycle notation, we have

$$\alpha = (1\ 2\ 3\ 4\ \cdots\ n{-}1).$$

So α is in fact a cycle of length $n-1$.

For β, notice that

$$\beta(i) = \begin{cases} b(i) & 1 \leq i \leq k, \\ ab(i) & k+1 \leq i \leq n-1, \end{cases}$$

which gives us

$$\beta = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & k & k{+}1 & \cdots & n{-}2 & n{-}1 \\ k{+}1 & 2 & 3 & 4 & \cdots & k & k{+}2 & \cdots & n{-}1 & 1 \end{pmatrix}.$$

Note that $\beta \in S_{n-1}$. In cycle notation, we have

$$\beta = (1\ k+1\ k+2\ \cdots\ n-1).$$

Now, if $k = 2$ we have

$$\begin{aligned} \beta\alpha^{-1} &= (1\ 3\ \cdots\ n-1)(1\ n-1\ \cdots\ 2) \\ &= (1\ 2). \end{aligned}$$

So $\operatorname{span}(\alpha, \beta) = S_{n-1}$. Otherwise l is even. Define

$$\begin{aligned} \beta' &= \alpha^{-1}\beta^2\alpha\beta^{-1} \\ &= (1\ n{-}1\ \cdots\ 2)(1\ k{+}1\ \cdots\ n{-}1)^2(1\ 2\cdots\ n{-}1)(1\ n{-}1\ \cdots\ k{+}1) \\ &= (1\ k{+}1)(2\ k{+}2\ \cdots\ n{-}1). \end{aligned}$$

It follows that $(\beta')^{l-1} = (1\ k{+}1)$. Hence $\operatorname{span}(\alpha, \beta) = S_{n-1}$.

In either case, we have that $\operatorname{span}\alpha, \beta = S_{n-1}$. It follows from Lemma 3.6 that $\operatorname{span}\alpha, \beta, \gamma = T_{n-1}$. □

Now let us characterize the elements of T_n that are not generated by $\{a, b\}$. These elements come in two basic forms described in the two following lemmas.

Define $A = \{1, \ldots, k\}$ and $B = \{k+1, \ldots, n\}$.

Lemma 3.21. *For $n \geq 5$, define*

$$O_{k,l} = \{q \in T_n : \text{all elements } B \text{ are in } \operatorname{img}(q)\}.$$

Then

$$U_{k,l} \cap\ O_{k,l} = \{a^i : i > 0\}.$$

Proof. Certainly $\{a^i : i > 0\} \subseteq O_{k,l}$ since $\text{img}(a^i) = Z_n$. If $w \notin \{a^i : i > 0\}$, then $w = w'ba^i$ for some $i \geq 0$ and some $w' \in U_{k,l}$. Let $j = a^i(n)$, then j is not in the image of w, since n is not in the image of b. Since $n \in B$, $j \in B$, and hence $w \notin O_{k,l}$. Therefore $U_{k,l} \cap O_{k,l} = \{a^i : i > 0\}$. □

Lemma 3.22. *For $n \geq 5$, define*

$$P_{k,l} = \{q \in T_n : q(A) \cap q(B) = \emptyset\}.$$

Then

$$U_{k,l} \cap P_{k,l} = \{a^i : i > 0\}.$$

Proof. Certainly $\{a^i : i > 0\} \subseteq P_{k,l}$ since a^i is a permutation. If $w \notin \{a^i : i > 0\}$, then $w = a^i bw'$ for some $i > 0$ and some $w' \in U_{k,l}$. Let s, t be such that $a^i(s) = 1$ and $a^i(t) = k + 1$. Then $a^i b(s) = a^i b(t)$ since $b(1) = b(k + 1)$. It follows that $w(s) = w(t)$. Furthermore, since $a^i(s) = 1$ and $a^i(t) = k + 1$, we must have that $s \in A$ and $t \in B$. Therefore $w \notin P_{k,l}$, so that $U_{k,l} \cap P_{k,l} = \{a^i : i > 0\}$. □

This next lemma shows that, with the exception of the permutations, the previously described elements are the only elements not generated by $\{a, b\}$.

Lemma 3.23. *For $n \geq 5$, let $w \in T_n$, with* $\text{rank}(w) < n$. *Then $w \in U_{k,l}$ if and only if $w \notin O_{k,l} \cup P_{k,l}$.*

Proof. If $w \in U_{k,l}$ has rank $< n$, then it follows from Lemma 3.21 and Lemma 3.22 that $w \notin O_{k,l} \cup P_{k,l}$.

For $w \notin O_{k,l} \cup P_{k,l}$, since $w \notin O_n$, there exists $r \in B$ such that $r \leq n$ and r is not in the range of w. Furthermore, since $w \notin P_{k,l}$ there exists $s \in A$ and $t \in B$ such that $w(s) = w(t)$. Let $i \geq 0$ be such that $a^i(s) = 1$ and $a^i(t) = k + 1$. Let $j \geq 0$ be such that $a^j(n) = r$. Then by Lemma 3.20, there exists $w' \in U_{k,l}$ be such that $a^i w' a^j = w$. This completes the proof. □

We are now ready to completely describe the monoid $U_{k,l}$.

Lemma 3.24. *For $n \geq 5$, we have*

$$U_{k,l} = \{a^i : i > 0\} \cup (T_n \setminus (O_{k,l} \cup P_{k,l})).$$

Proof. It is easy to see that

$$U_{k,l} \cap S_n = \{a^i : i > 0\}.$$

Certainly $S_n \subset O_n$. Then it follows from Lemma 3.23, that

$$U_{k,l} \backslash S_n = T_n \backslash (O_{k,l} \cup P_{k,l}).$$

Hence

$$U_{k,l} = \{a^i : i > 0\} \cup (T_n \backslash (O_{k,l} \cup P_{k,l})).$$

□

For any function $f : Z_n \to Z_n$ in T_n, we can view the elements of Z_n, in the domain, as vertices of a graph, and the elements of Z_n, in the range, as colours. This provides a correspondence between transformations and vertex colourings of a graph. By viewing T_n in this way, we see that each function in $P_{k,l}$ corresponds to a colouring of a graph with vertex set Z_n, such that no vertex in A has the same colour as a vertex in B. Then the number of elements in $P_{k,l}$ is exactly the number of proper vertex colourings of the complete bipartite graph, where one component is the set of vertices A, and the other component is the set of vertices B.

Lemma 3.25. *For positive integers k,l, and j, define $g(k, l, j)$ to be the number of proper vertex colourings, using j colours, of the complete bipartite graph with vertex sets of size k and l. Then for $n \geq 5$ there exist coprime integers k and l such that $k + l = n$, and*

$$g(k, l, n) \leq n^n \left(\frac{4(n-1)}{n^2 + 4n - 20} \right).$$

Proof. If n odd, take $k = \frac{n-1}{2}$, and $l = \frac{n+1}{2}$. Otherwise, if $n = 2i$ for some even i, take $k = \frac{n}{2} - 1$, and $l = \frac{n}{2} + 1$. Otherwise, $n = 2i$ for some odd i, so take $k = \frac{n}{2} - 2$, and $l = \frac{n}{2} + 2$. We can easily verify that, in all cases, the choice of k, l is coprime.

We can give an upper bound on the size of $g(k, l, j)$ using the following result, proved by Lazebnik [16], concerning the number of proper colourings of a graph.

Theorem 3.26 (Lazebnik). *Let v, e, and λ be integers with $0 \leq e \leq v(v-1)/2$, and $\lambda \geq 2$. Define $\kappa(v, e, \lambda)$ to be the greatest number of proper colourings of a graph with v*

vertices and e edges using λ colours. Then

$$\kappa(v, e, \lambda) \leq \left(\frac{\lambda - 1}{\lambda - 1 + e}\right) \lambda^v.$$

There are kl edges in a complete bipartite graph with vertex sets of size k and l. If we take $\lambda = j$, $v = n$, and $e = kl$, then by Theorem 3.26 we have

$$g(k, l, j) \leq \frac{(j-1)j^n}{j - 1 + kl} \ .$$

But $\left(\frac{n}{2} + 2\right) \cdot \left(\frac{n}{2} - 2\right) \leq kl$ so that

$$\begin{aligned} g(k, l, n) &\leq \frac{(j-1)j^n}{j - 1 + \left(\frac{n}{2} + 2\right) \cdot \left(\frac{n}{2} - 2\right)} \\ &\leq \frac{(j-1)k^n}{j - 1 + \frac{(n-1)^2}{4}} \\ &\leq n^n \left(\frac{4(n-1)}{n^2 + 4n - 20}\right), \end{aligned}$$

as required. □

The connection between the elements of $O_{k,l}$ and the strings described in the following lemma is clear.

Lemma 3.27. *Let $n = k + l$, with $k < l$. Define $h(k, l, j)$ to be the number of length-j strings over the alphabet $\{1, \ldots, n\}$, such that each string contains at least one occurrence of each of the elements of $\{k + 1, \ldots, n\}$. Then*

$$h(k, l, n) < 1.5372 \cdot n^{n+4} \left(\frac{2}{e}\right)^{\frac{n}{2}}.$$

Proof. There are l elements in the set $\{k+1, \ldots, n\}$. We can select the l positions for these elements in $\binom{n}{l}$ ways. For each selection of positions, we can arrange the odd elements in $l!$ ways. For each arrangement, the remaining k positions can be occupied in no more than

n^k ways. Then we have

$$\begin{aligned} h(n,n) &\leq \binom{n}{l} \cdot l! \cdot n^k \\ &= \frac{n!}{l!} \cdot (l)(l-1)\cdots(k+1) \cdot n^k \\ &< \frac{n!}{l!} \cdot n^{k+4} \end{aligned}$$

A version of Stirling's Formula [30] gives us

$$\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \cdot e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \cdot e^{\frac{1}{12n}}.$$

For $n \geq 1$, $e^{\frac{1}{12n+1}} > 1$, and $e^{\frac{1}{12n}} \leq e^{\frac{1}{12}}$. So

$$\frac{n!}{l!} < \frac{n^{n+\frac{1}{2}} \cdot e^{-n} \cdot e^{\frac{1}{12}}}{l^{l+\frac{1}{2}} \cdot e^{-l}}.$$

But since $x^{x+\frac{1}{2}}e^{-x}$ is increasing for $x > 0$, we have

$$\begin{aligned} \frac{n!}{l!} &< \frac{n^{n+\frac{1}{2}} \cdot e^{-n} \cdot e^{\frac{1}{12}}}{\left(\frac{n}{2}\right)^{\frac{n}{2}+\frac{1}{2}} \cdot e^{-\frac{n}{2}}} \\ &= e^{\frac{1}{12}} \cdot n^{\frac{n}{2}} \cdot 2^{\frac{n}{2}+\frac{1}{2}} \cdot e^{-\frac{n}{2}} \\ &< 1.5372 \cdot n^{\frac{n}{2}} \left(\frac{2}{e}\right)^{\frac{n}{2}}, \end{aligned}$$

where $1.5372 \doteq \sqrt{2}e^{\frac{1}{12}}$. This gives us the desired result. □

We are now ready to give a lower bound on the size of the largest monoid generated by two transformations.

Theorem 3.28. *Define A_n to be the largest submonoid of T_n that can be generated by two elements. Then for $n \geq 189$ we have*

$$|A_n| > n^n(1 - 4n^{-1}).$$

Proof. Choose coprime integers k, l with k as large as possible, so that $k + l = n$, and $1 < k < l < n$. Certainly $|A_n| \geq |U_{k,l}|$. Define $g(k, l, n)$ as in Lemma 3.25. Then $|P_{k,l}| = g(n, n)$. Also, define $h(k, l, n)$ as in Lemma 3.27, so that $|O_{k,l}| = h(k, l, n)$. It follows from Lemma 3.24, that

$$|U_{k,l}| = n^n - |O_{k,l}| - |P_{k,l}| + |O_{k,l} \cap P_{k,l}| + kl.$$

This gives us

$$\begin{aligned}
|A_n| &\geq |U_{k,l}| \\
&= n^n - |O_{k,l}| - |P_{k,l}| + |O_{k,l} \cap P_{k,l}| + k \cdot l \\
&> n^n - |O_{k,l}| - |P_{k,l}| \\
&= n^n - h(k, l, n) - g(k, l, n) \\
&> n^n - 1.5372 \cdot n^{n+4} \left(\frac{2}{e}\right)^{\frac{1}{2}n} - n^n \left(\frac{4(n-1)}{n^2 + 4n - 20}\right) \qquad \text{(by Lemma 3.27 and Lemma 3.25).}
\end{aligned}$$

The reader can verify that for $n \geq 189$, we have $\frac{4}{n} > \frac{4(n-1)}{n^2+4n-20} + 1.5372 \cdot n^4 \left(\frac{2}{e}\right)^{\frac{1}{2}n}$, which gives

$$|A_n| > n^n(1 - 4n^{-1}).$$

□

Corollary 3.29. *For each $n \geq 5$, there exists a monoid A_n such that*

$$\lim_{n\to\infty} \frac{|A_n|}{n^n} = 1.$$

Proof. The result is an immediate consequence of Theorem 3.28. □

This work was completed in May 2003 by Krawetz, Lawrence, and Shallit [13]. Unfortunately, these results were already known to Holzer and König [9].

We now give a summary of the proof, due to Holzer and König, that the $U_{k,l}$ monoids are, in fact, the largest two-generated monoids. However, for the purposes of their use in the next chapter, many of the intermediate results given by Holzer and König have been improved upon, somewhat. In these cases, the proofs are very similar to those given for

the original results, and include only small changes. In each case, the statement of the original result will follow as a corollary.

We begin with a slightly more general definition of the $U_{k,l}$ monoids, as used by Holzer and König.

For coprime integers $k, l \geq 2$, where $k+l = n$, let $\alpha = (1\ 2\ \cdots\ k)(k+1\ \ k+2\ \ \cdots\ n)$ be a permutation of Z_n composed of two cycles, one of length k, the other of length l. Define $U_{k,l}$ to be the set of all transformations $\gamma \in T_n$ where exactly one of the following is true:

1. $\gamma = \alpha^m$ for some positive integer m;
2. For some $i \in \{1, \cdots, k\}$ and some $j \in \{k+1, \cdots, n\}$ we have that $\gamma(i) = \gamma(k)$ and for some $m \in \{k+1, \cdots, n\}$ we have that $m \notin \text{img}(\gamma)$.

Let $\pi_1 = (1\ 2\ \cdots\ k)$ be an element of S_{n-1}, and let $\pi_2 \in S_{n-1}$ be a permutation such that π_1 and π_2 generate S_{n-1}. Now define $\beta \in T_n$ by

$$\beta = \begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ \pi_2(1) & \pi_2(2) & \cdots & \pi_2(n-1) & \pi_2(1) \end{pmatrix}.$$

Lemma 3.30 (Holzer and König). *The set $U_{k,l}$ is a submonoid of T_n and is generated by $\{\alpha, \beta\}$.*

Holzer and König [10] also gave the following formula to compute the size of $U_{k,l}$.

Lemma 3.31 (Holzer and König). *For $n = k + l$, we have*

$$|U_{k,l}| = kl + \sum_{i=1}^{n} \left(\binom{n}{i} - \binom{k}{i-l} \right) \left(\left\{ {n \atop i} \right\} - \sum_{r=1}^{i} \left\{ {k \atop r} \right\} \left\{ {l \atop i-r} \right\} \right) i!,$$

where $\left\{ {n \atop i} \right\}$ is a Stirling number of the second kind, the number of ways to partition a set of n elements into i non-empty sets.

Theorem 3.32 (Holzer and König). *For $n \geq 7$, there exist coprime integers k,l such that $n = k + l$ and*

$$|U_{k,l}| \geq n^n \left(1 - \sqrt{2} \left(\frac{2}{e} \right)^{\frac{n}{2}} e^{\frac{1}{12}} - \sqrt{8} \frac{1}{\sqrt{n}} e^{\frac{1}{12}} \right).$$

In the previous section, we defined the monoid V_n^l. This monoid is generated by a single-cycle permutation, and a non-bijective transformation that identifies two elements from that cycle. The next lemma gives an upper bound on the size of a monoid that is more general than V_n^l.

Lemma 3.33 (Holzer and König). *Let $a \in T_n$ be any permutation, and let $b \in T_n$ be a non-bijective transformation that identifies two elements of the same cycle in a. Then*

$$|\operatorname{span}(a,b)| \leq n^n + \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12}} - n^n\left(\frac{(n-1)(n-2)}{n^2}\right)^{\frac{n}{3}}.$$

With Lemma 3.32 and Lemma 3.33, we are now ready to give the first result leading to the maximality of the monoid $U_{k,l}$.

Lemma 3.34. *Let $a \in T_n$ be any permutation, and let $b \in T_n$ be a non-bijective transformation that identifies two elements of the same cycle in a. Then there exist coprime integers k,l, with $n = k + l$, such that*

$$|U_{k,l}| \geq |\operatorname{span}(a,b)| + \binom{n}{2}.$$

Proof. From Lemma 3.32 it follows that for some coprime integers k,l, with $n = k + l$, we have

$$|U_{k,l}| \geq n^n\left(1 - \sqrt{2}\left(\frac{2}{e}\right)^{\frac{n}{2}} e^{\frac{1}{12}} - \sqrt{8}\frac{1}{\sqrt{n}}e^{\frac{1}{12}}\right).$$

Also, from Lemma 3.33, we have

$$|\operatorname{span}(a,b)| \leq n^n + \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12}} - n^n\left(\frac{(n-1)(n-2)}{n^2}\right)^{\frac{n}{3}}.$$

Then $|\operatorname{span}(a,b)| \leq |U_{k,l}|$ when

$$\underbrace{\sqrt{2}\left(\frac{2}{e}\right)^{\frac{n}{2}} e^{\frac{1}{12}}}_{A(n)} + \underbrace{\sqrt{8}\frac{1}{\sqrt{n}}e^{\frac{1}{12}}}_{B(n)} < \underbrace{\left(\frac{(n-1)(n-2)}{n^2}\right)^{\frac{n}{3}}}_{C(n)} - \underbrace{\sqrt{2\pi n}\left(\frac{1}{e}\right)^n e^{\frac{1}{12}}}_{D(n)}.$$

For $n = 20$, we have $C(n) > 0.358$ and $D(n) < 10^{-7}$. On the interval $[20, \infty)$, the function $C(n)$ is strictly increasing while the function $|D(n)|$ is strictly decreasing. So for

$n \geq 20$, this gives $C(n) - D(n) > 0.357$. For $n = 82$, we have $A(n) < 10^{-4}$ and $B(n) < 0.34$. On the interval $[82, \infty)$, both $A(n)$ and $B(n)$ are strictly decreasing. So for $n \geq 82$, this gives $A(n) + B(n) < 0.35$. Hence, for $n \geq 82$, we have $C(n) - D(n) - A(n) - B(n) > 0.007 > n^{2-n}$. This gives

$$|U_{k,l}| - |\operatorname{span}(a, b)| \geq n^{2-n} \cdot n^n = n^2 > \binom{n}{2}.$$

For $n \leq 81$, Holzer and König [10] verified computationally that the largest such $\operatorname{span}(a, b)$ monoids were, in fact, the V_n^l monoids. This author verified computationally that for $7 \leq n \leq 81$ and some appropriate k,l, we have $|U_{k,l}| \geq |V_n^j| + \binom{n}{2}$ for all j. □

Corollary 3.35 (Holzer and König). *Let $a \in T_n$ be any permutation, and let $b \in T_n$ be a non-bijective transformation that identifies two elements of the same cycle in a. Then there exist coprime integers k,l, with $n = k + l$, such that*

$$|U_{k,l}| \geq |\operatorname{span}(a, b)|.$$

Lemma 3.36. *For prime numbers $n \geq 7$, let $\alpha \in S_n$ be a permutation with exactly 2 cycles, and let $\beta \in T_n$ be any transformation that identifies elements from at least two different cycles of α. Let V denote* $\operatorname{span}(\alpha, \beta)$. *Then either V is isomorphic to a $U_{k,l}$ monoid, or there exist coprime integers k,$l \geq 2$ such that $k + l = n$ and*

$$|U_{k,l}| \geq |V| + \binom{n}{2}.$$

Proof. Let s and t denote the lengths of the the two cycles in α, and let r denote the element missing from the image of β.

In the degenerate case, where one of the cycles in α has length 1, it follows from Lemma 3.16 and Lemma 3.34 that for some appropriate k, l we have

$$|U_{k,l}| \geq |V| + \binom{n}{2}.$$

When α has no cycle of length 1, we have $s, t \geq 2$. let C_s be the set of elements contained in the cycle of length s, and let C_t be defined similarly. Without loss of generality, assume $r \in C_s$. Let

$$V' = \operatorname{span}(\alpha) \cup \{\gamma \in T_n : m \notin \operatorname{img}(\gamma),\ m \in C_s;\ \text{and } \gamma(i) = \gamma(j),\ i \in C_s, j \in C_t\}.$$

Certainly $V \subseteq V'$. If $V = V'$, then V is isomorphic to $U_{s,t}$.

Now, if $V \subset V'$, then assume that $r = n$, and let V'' denote the restriction of V to the domain and range Z_{n-1}. Then $V'' \neq T_{n-1}$, otherwise, an argument similar to the one used in the proof of Lemma 3.24 would show that $V = V'$. Hence, there exists $X \subseteq T_{n-1}$, such that $V'' \subseteq T_{n-1} \backslash X$. It follows from arguments similar to those used in the proof of Lemma 3.7 that X contains at least all transpositions, all $(n-1)$-cycles, or all $(n-1)$-rank transformations.

If X contains all transpositions, then $|X| \geq \binom{n-1}{2}$. Furthermore, for every $\gamma \in X$, let $\gamma' \in T_n$ be an extension of γ such that $\gamma' \in V'$. Then $\gamma'(n)$ can be any element of C_t. But $\gamma' \notin V$. Hence,

$$|V| \leq |V'| - t \cdot \binom{n-1}{2} \leq |U_{s,t}| - \binom{n}{2}.$$

If X contains all $(n-1)$-cycles, then a similar argument shows that

$$|V| \leq |V'| - t \cdot (n-2)! \leq |U_{s,t}| - \binom{n}{2}.$$

Finally, if X contains all $(n-1)$-rank transformations, then $|X| \geq \binom{n-1}{2}(n-2)!$, so that

$$|V| \leq |V'| - \binom{n-1}{2}(n-2)! \leq |U_{s,t}| - \binom{n}{2}.$$

If $r \neq n$, then a similar argument, considering the restriction of V to the domain and range $Z_n \backslash \{r\}$, will lead to the same conclusion. □

Lemma 3.37. *For prime numbers $n \geq 7$, let $\alpha \in S_n$ be any permutation with at least 3 cycles, and let $\beta \in T_n$ be any transformation that identifies elements from at least two different cycles of a. Then there exist coprime integers k,$l \geq 2$ such that $k + l = n$ and*

$$|U_{k,l}| \geq |\operatorname{span}(\alpha, \beta)| + \binom{n}{2}.$$

Proof. Let U denote the monoid $\operatorname{span}(\alpha, \beta)$. We will construct a new monoid U' that is isomorphic to some $U_{k,l}$ monoid, and show that $|U'| \geq |U| + \binom{n}{2}$.

Define i, j to be elements from two different cycles of α such that $\beta(i) = \beta(j)$. Furthermore, define r to be an element missing from the image of β.

Let m be the number of cycles in α, and let $k_1, \ldots, k_m$ be their lengths. For $1 \leq i \leq m$, define C_i as the set of elements contained in the cycle of length k_i. Then

$$\sum_{1 \leq i \leq m} k_i = n, \text{ and } \bigcup_{1 \leq i \leq m} C_i = Z_n.$$

Without loss of generality, assume that $i \in C_1$ and $j \in C_2$. Then either $r \in C_1 \cup C_2$, or $r \notin C_1 \cup C_2$.

Case 1: $r \in C_1 \cup C_2$.

Without loss of generality, assume that $r \in C_2$. Let α' be a permutation composed of two cycles; the first containing the elements $C_1' = C_1$, and the second containing the elements $C_2' = C_2 \cup \cdots\cdots C_m$ Let $k = |C_1'|$, and $l = |C_2'|$.

Since $n = k + l$ is prime, we have that k and l are coprime, and we can find a transformation β' so that $\text{span}(\alpha', \beta') = U'$, where

$$U' = \text{span}(\alpha') \cup \{\gamma \in T_n : m \notin \text{img}(\gamma),\ m \in C_1';\ \text{and}\ \gamma(i) = \gamma(j),\ i \in C_1', j \in C_2'\},$$

with U' isomorphic to $U_{k,l}$. Now,

$$U \subseteq \text{span}(\alpha) \cup \{\gamma \in T_n : m \notin \text{img}(\gamma),\ m \in C_1;\ \text{and}\ \gamma(i) = \gamma(j),\ i \in C_1, j \in C_2\}.$$

It follows then, since $C_1 \subseteq C_1'$, and $C_1 \subseteq C_1'$, that $U \backslash S_n \subseteq U' \backslash S_n$.

In the worst case the monoid U may contain $k_1 \cdots k_m$ permutations, while U' contains exactly kl. However, we will show that the difference in the number of rank $n-1$ transformations is more than enough to compensate.

We enumerate the rank $n-1$ elements of U as follows: select an element of C_1 to identify with an element for C_2 (this can be done in $k_1 k_2$ ways); select an element of C_2 to omit from the image (this can be done in k_2 ways); finally, arrange the elements of the image, in $(n-1)!$ ways. Then we have at most $k_1 k_2^2 (n-1)!$ transformations of rank $n-1$ in the monoid U. By similar reasoning, there are exactly $kl^2(n-1)!$ transformations of rank $n-1$ in the monoid U'. This gives a difference of $(kl^2 - k_1 k_2^2)(n-1)!$.

Case 1.a: $k_1 \geq \frac{n}{2} + 1$.

We have,

$$\begin{aligned} kl^2 - k_1k_2^2 &= k_1(l^2 - k_2^2) \\ &= k_1(l + k_2)(l - k_2) \\ &\geq \left(\frac{n}{2} + 1\right)(l + k_2)(l - k_2) \\ &\geq \left(\frac{n}{2} + 1\right)(2) \quad \text{since } 1 \leq k_2 < l \\ &\geq n + 1. \end{aligned}$$

Case 1.b: $k_1 < \frac{n}{2} + 1$.

Since $k_1 < \frac{n}{2} + 1$, we have that $l \geq \frac{n}{2} + 1$. Now, since $1 \leq k_2 < l$, this also gives us $(l + k_2) \geq n + 1$, or $(l - k_2) \geq 2$. Thus,

$$\begin{aligned} kl^2 - k_1k_2^2 &= k_1(l^2 - k_2^2) \\ &\geq (l + k_2)(l - k_2) \\ &\geq n + 1. \end{aligned}$$

In either case, this gives $(n + 1)(n - 1)!$ more elements in $U'\backslash S_n$. Therefore

$$\begin{aligned} |U'\backslash S_n| &\geq |U\backslash S_n| + (n + 1)(n - 1)! \\ &\geq |U\backslash S_n| + n! + \binom{n}{2} \\ &\geq |U| + \binom{n}{2}. \end{aligned}$$

Case 2: $r \notin C_1 \cup C_2$.

Without loss of generality, assume that $r \in C_3$. Let α' be a permutation composed of two cycles; the first containing the elements $C_1' = C_1 \cup \bigcup_{4 \leq i \leq m} C_i$, and the second containing the elements $C_2' = C_2 \cup C_3$ Let $k = |C_1'|$, and $l = |C_2'|$. Then $kl^2 - k_1k_2k_3 \geq n + 1$. So by an argument similar to the one given for case 1, we have the desired result.

Hence, the result is proved. □

Corollary 3.38 (Holzer and König). *For prime numbers $n \geq 7$, let $\alpha \in S_n$ be any permutation with at least 2 cycles, and let $\beta \in T_n$ be any transformation that identifies elements from at least two different cycles of a. Then there exist coprime integers $k,l \geq 2$ such that $k + l = n$ and*

$$|U_{k,l}| \geq |\operatorname{span}(\alpha, \beta)|.$$

We are now ready to state the main result of this chapter.

Theorem 3.39. *For all prime numbers $n \geq 7$, there exist coprime integers $k,l \geq 2$ such that $k + l = n$ and for any two-generated monoid $U \subseteq T_n$, we have $|U_{k,l}| \geq |U| + \binom{n}{2}$.*

Proof. If U is a two-generated monoid, then we must have one of the following cases:
Case 1: U is generated by two permutations.

This case is covered by Lemma 3.17 and Lemma 3.34.

Case 2: U is generated by two non-bijective functions.

This case is covered by Lemma 3.18 and Lemma 3.34.

Case 3: U is generated by a permutation and a non-bijective function identifying two elements from the same cycle.

This case is covered by Lemma 3.16 and Lemma 3.34.

Case 4: U is generated by a permutation and a non-bijective function identifying two elements from different cycles.

This case is covered by Lemma 3.36 and Lemma 3.37.

Hence, we have the desired result. □

Corollary 3.40 (Holzer and König). *For all prime numbers $n \geq 7$, there exist coprime integers $k,l \geq 2$ such that $k + l = n$ and $U_{k,l}$ is the largest two-generated submonoid of T_n.*

It seems likely that the results of Theorem 3.39 and Corollary 3.40, will hold in general and so we give the following conjectures.

Conjecture 3.41 (Holzer and König). *For any $n \geq 7$, there exist coprime integers $k,l \geq 2$ such that $k + l = n$ and $U_{k,l}$ is the largest two-generated submonoid of T_n.*

Conjecture 3.42. *For any $n \geq 7$, there exist coprime integers $k,l \geq 2$ such that $k + l = n$ and for any two-generated monoid $U \subseteq T_n$, we have $|U_{k,l}| \geq |U| + \binom{n}{2}$.*

Though Corollary 3.40 gives us proof that a $U_{k,l}$ monoid is maximal when n is prime, it does not offer any method to determine the integers k and l. Holzer and König believe, however, that these values are as close to $\frac{n}{2}$ as possible, as in the proof of Lemma 3.25.

Chapter 4

State complexity of root(L)

In this chapter, we focus on determining the worst-case state complexity of the operation root(L), given by

$$\text{root}(L) = \{w \in \Sigma^* : \exists n \geq 1 \text{ such that } w^n \in L\},$$

where L is a regular language. Note that this operation is not the same as the ROOT(L) operation studied by Horváth, Leupold, and Lischke [12].

The study of the root(L) operation requires us to examine the connections between finite automata and algebra. We will show that the algebraic structure of a DFA recognizing L provides a basis for the construction of a DFA recognizing root(L). This relationship allows us to apply the results of Chapter 3 to obtain non-trivial lower bounds on the worst-case state complexity of root(L).

The operation root(L) is regularity preserving, that is, when $L \subseteq \Sigma^*$ is regular, root(L) is regular. This fact can be demonstrated by applying Theorem 2.1 of [27], with $\lambda = [1], \nu = [1]$, and $\mu : \Sigma^* \to \mathcal{P}(\Sigma^*)^{1\times 1}$ given by $\mu(w) = [w^+]$. Here $\mathcal{P}(\Sigma^*)^{1\times 1}$ is the set of 1×1 matrices with entries in the power set of Σ^*, and w^+ is the set $\{w, w^2, w^3, \ldots\}$.

For the purposes of this thesis, however, we require a construction for root(L) involving automata. The first construction we give uses a type of automaton known as a 2DFA. The class of languages recognized by these automata was shown, by Rabin and Scott [28], to be equivalent to the class of regular languages. Here we will give a definition similar to that used by Birget [4].

Intuitively, a DFA is a machine that processes its input tape. At any given point during processing, the tape head of the DFA is located at some position on the input tape. After each step of processing, the tape head moves one symbol to the right. In this sense, a DFA is only capable of reading its input tape once, from left to right. A 2DFA, on the other hand, is an extension of this model. This machine, instead, is permitted to move its tape head left, right, or not at all. So a 2DFA is capable of reading its input tape an arbitrary number of times.

More formally, a *two-way deterministic finite automaton*, or *2DFA*, is is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite non-empty set of states, Σ is the finite input alphabet, $\delta : Q \times \Sigma \cup \{\#, \$\} \to Q \times \{-1, 0, 1\}$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. Note that the symbols $\#$ and $\$$ are not in Σ. Instead, they serve as left and right tape end-markers, respectively.

A *configuration* of $\mathcal{A}$ is composed of a state (considered the *current* state), the contents of the input tape, and the position of the tape head on the tape. On the input $w \in \Sigma^*$, a configuration can be represented by a string of the form $w'qxw''$, where $q \in Q$ is the current state, $x \in \Sigma \cup \{\#, \$\}$, and $w'xw'' = \#w\$$. In the configuration $w'qxw''$, the tape head is located at the symbol x. The automaton $\mathcal{A}$ begins computation in the configuration represented by $q_0\#w\$$. For any configuration of the form $w'qxw''$, let $\delta(q, x) = (q', d)$. If $\delta(q, x)$ is not defined, or if $w' = \epsilon$ and $d = -1$, then the automaton *crashes* and processing stops. Otherwise, the next configuration is exactly one of the following:

1. $w'xq'w''$, if $d = 1$;
2. $w'q'xw''$, if $d = 0$;
3. $v'qx'v''$, where $v'x' = w'$ and $v'' = xw''$, if $d = -1$.

Then the computation of a word $w \in \Sigma^*$ is a sequence of configurations beginning with $q_0\#w\$$. If the configuration $w'qw''$ is precedes the configuration $v'pv''$ in the sequence, then we say that $v'pv''$ is *reachable* from $w'qw''$ and we write $w'qw'' \Rightarrow^* v'pv''$. If $q_0\#w\$$ reaches the configuration $\#w\$q$, then processing stops. If $q \in F$, then we say that $\mathcal{A}$ *accepts* or *recognizes* w. The language recognized by $\mathcal{A}$ is given by

$$L(\mathcal{A}) = \{w \in \Sigma^* \ : \ w \text{ is accepted by } \mathcal{A}\} \subseteq \Sigma^*.$$

Note that it is possible for a machine to reach the same configuration more than once. In this case, the automaton has entered an infinite loop, and processing will never stop.

The definition given above for a 2DFA includes the use of end-markers. Though this use is common, it is not strictly necessary. Shepherdson [33] proved that end-markers provide no additional computational power. That is, the class of languages recognizable by 2DFA does not depend on the use of end-markers in the definition.

Theorem 4.1. *The operation* $\text{root}(L)$ *is regularity preserving.*

Proof. The idea of this construction is to use a 2DFA to simulate the behaviour of a DFA recognizing L. However, when the end of input is reached, computation in the DFA is suspended while the tape head is brought back to the beginning of the input string. Once the tape head is reset, computation resumes. If, when the end of input is reached, the 2DFA is in a final state of the simulated DFA, then the machine has done so by reading a string of the form w^i. In this case, the 2DFA accepts the string.

More formally, for a regular language L, let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing L. Define the 2DFA $\mathcal{A}' = (Q', \Sigma, \rho, q_0', F')$ as follows:

- $Q' = Q \cup \{q' : q \in Q\}$;
- $F' = \{q' \in Q' : q \in F\}$;
- $\rho(q, a) = (\delta(q, a), 1)$, for all $q \in Q$ and $a \in \Sigma$;
- $\rho(q, \$) = (q, -1)$, for all $q \in Q$, $q \notin F$;
- $\rho(q, \$) = (q', 1)$, for all $q \in Q$, $q \in F$.
- $\rho(q', a) = (q', -1)$, for all $q \in Q$ and $a \in \Sigma$;
- $\rho(q', \#) = (q, 1)$, for all $q \in Q$;

To see that $L(\mathcal{A}') = \text{root}(L)$, consider any $w \in \text{root}(L)$. Let $k \geq 1$ be the least integer such that $w^k \in L$. Let denote $q_i = \delta(q_0, w^i)$ for $1 \leq 1 \leq k$. Then $q_k \in F$, since $w^k \in L$. This gives us

$$q_0'\#w\$ \Rightarrow^* \#q_0w\$ \Rightarrow^* \#wq_1\$ \Rightarrow^* q_1'\#w\$ \Rightarrow^* \#q_1w\$ \Rightarrow^* \#wq_k\$ \Rightarrow^* \#w\$q_k'.$$

Hence, $w \in L(\mathcal{A}')$.

Now, while processing $w \in \Sigma^*$ it is clear that the movement of the tape head will only change direction when a tape end-marker is reached. So we can consider the configuration of $\mathcal{A}'$ after each left-to-right scan of the string w. Let q_i denote the current state of the the 2DFA after the ith scan, when the tape head is on the right end-marker. Then the movement of the tape head back to the beginning of input is represented by the sequence $\#wq_i\$ \Rightarrow^* \#q_iw\$$, so q_i is also the current state at the beginning of the $(i+1)$th scan.

If $w \in L(\mathcal{A}')$, then w is scanned k times, where k is an integer. Furthermore, we have $q_k \in F'$. Each scan of the input gives us $\delta(q_i, w) = q_{i+1}$ for $1 \leq i < k$. Then it follows that

$$\delta(q_0, w^k) = \delta(q_1, w^{k-1}) = \delta(q_{k-1}, w) = q_k \in F.$$

Hence $w \in \text{root}(L)$. This completes the proof. □

Though the idea behind the machine constructed in the proof of Theorem 4.1 is simple, the notation is cumbersome. This makes the execution of the idea much more complicated than desired. A more elegant proof, involving automata, of the regularity of root(L) is given in the following theorem.

Theorem 4.2. *For a language L and a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $L = L(\mathcal{A})$, define the DFA $\mathcal{A}' = (Q^Q, \Sigma, \delta', q_0', F')$ where $q_0' = \delta_\epsilon$, $F' = \{f : \exists n \geq 1$ such that $f^n(q_0) \in F\}$, and δ' is given by*

$$\delta'(f, a) = f\delta_a, \text{ for all } f \in Q^Q \text{ and } a \in \Sigma.$$

Then root(L) $= L(\mathcal{A}')$.

Proof. An easy induction on $|w|$, $w \in \Sigma^*$, proves that $\delta'(q_0', w) = \delta_w$. Then

$$\begin{aligned} x \in \text{root}(L) &\Leftrightarrow \exists n \geq 1 \ : \ x^n \in L \\ &\Leftrightarrow \delta_x(q_0) \in F' \\ &\Leftrightarrow \delta'(q_0', x) \in F'. \end{aligned}$$

□

The construction of Theorem 4.2 uses, as its states, transformations of states of the automaton for L. Zhang [38] used a similar technique to characterize regularity-preserving operations. To recognize the image of a language under an operation, Zhang constructed a new automaton with states based on Boolean matrices. These matrices represent the transformations of states in the original automaton.

The result of Theorem 4.2 also allows us to give our first bound on the state complexity of root(L).

Corollary 4.3. *For regular language L, if* $\mathrm{sc}(L) = n$ *then* $\mathrm{sc}(\mathrm{root}(L)) \leq n^n$.

Proof. This is immediate from the construction given in Theorem 4.2. □

Despite the simplicity of its description, the DFA in Theorem 4.2 is actually quite large, using n^n states versus the $2n$ states used in the 2DFA construction of Theorem 4.1. This suggests at least an $n^{O(n)}$ upper bound on the state complexity of a language recognized by an n-state 2DFA. We will revisit this idea in the last section of this chapter. For now, we continue with an analysis of the state complexity of root(L).

4.1 Unary languages

For unary regular languages, the problem is simple enough that we are able to attack it directly, without having to employ results from algebra. In the unary case, it turns out that the state complexity of the root of a language is bounded by the state complexity of the original language.

Proposition 4.4. *If L is a unary regular language, then* $\mathrm{sc}(\mathrm{root}(L)) \leq \mathrm{sc}(L)$. *This bound is tight.*

The idea of the following proof is that given a particular DFA recognizing L, we can modify it by adding states to the set of final states. The resulting DFA will recognize the language root(L).

Proof. Let $\Sigma = \{a\}$ be the alphabet of L. Since L is regular and unary, there exists a DFA $\mathcal{A}$ recognizing L, such that $\mathcal{A} = (\{q_0, \ldots, q_{n-1}\}, \{a\}, \delta, q_0, F)$, where

$$\delta(q_i, a) = q_{i+1}, \quad \text{for all } 0 \leq i < n-1,$$

and

$$\delta(q_{n-1}, a) = q_j, \quad \text{for some } 0 \leq j \leq n-1.$$

We call the states $q_0, \ldots, q_{j-1}$ the *tail*, and the states $q_j, \ldots, q_{n-1}$ the *loop*.

Notice that $\text{root}(L) = \{a^s \in \Sigma^* : s \mid t, a^t \in L\}$. Since words in $\text{root}(L)$ are divisors of words in L, the following theorem [2, Thm. 4.3.1] is useful.

Theorem 4.5. *Let u, v, and f be integers. Then the equation*

$$gu + hv = f$$

has a solution in integers g and h if and only if $\gcd(u, v) \mid f$.

For all strings $a^t \in L$, we have some $k \geq 0$ and some $b \leq n-1$ such that $t = kl + b$, where $l = n - j$ is the number of states in the loop. Let $s = lm + c$ for some $m \geq 0$ and some $0 \leq c < l$. Then

$$\begin{aligned}
s \mid t &\Leftrightarrow \exists r \ : \ lk + b = r(lm + c) \\
&\Leftrightarrow \exists r \ : \ lk - rlm = rc - b \\
&\Leftrightarrow \exists r \ : \ \gcd(l, -lm) \mid rc - b && \text{(by Theorem 4.5)} \\
&\Leftrightarrow \exists r \ : \ l \mid rc - b \\
&\Leftrightarrow \exists r, v \ : \ rc - b = lv \\
&\Leftrightarrow \exists r, v \ : \ rc - lv = b \\
&\Leftrightarrow \gcd(l, c) \mid b. && \text{(by Theorem 4.5)}
\end{aligned}$$

It follows that the set of divisors of a number of the form $kl + b$, $k \geq 0$, $b \leq n-1$ is as follows:

$$\{lm + c \in \mathbb{Z} : m \geq 0, \gcd(l, c) \mid b\}.$$

Then for each $a^t \in L$, the divisors of $t = kl + b$ can be recognized by changing the corresponding states into final states. Therefore, $\text{sc}(\text{root}(L)) \leq \text{sc}(L)$.

To show that this bound is tight, for $n \geq 2$ consider the language $L_n = \{a^{n-2}\}$. Under the Myhill–Nerode equivalence relation $\sim_{L_n}$, no two strings in the set $\{\epsilon, a, a^2, \ldots, a^{n-2}\}$ are equivalent. That is, for any two strings a^i and a^j, where $0 \leq i, j \leq n-2$, we can

distinguish a^i and a^j with the string a^{n-2-i}. All other strings in Σ^* are equivalent to a^{n-1}. This gives $\mathrm{sc}(L_n) = n$. Furthermore, since a^{n-2} is the longest word in $\mathrm{root}(L_n)$, $\delta(q_0, a^{n-2})$ cannot be a state in the loop. It follows that we require exactly $n-1$ states in the tail plus a single, non-final state in the loop. Hence $\mathrm{sc}(\mathrm{root}(L_n)) = n$. Therefore the bound is tight. □

4.2 Languages on larger alphabets

First, recall, from the beginning of Chapter 3, the definition of the transition monoid of an automaton.

Definition. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be DFA. For $w \in \Sigma^*$, define the function $\delta_w : Q \to Q$ by $\delta_w(q) = \delta(q, w)$, for all $q \in Q$. If we denote the empty word by ϵ, then δ_ϵ is the identity function. Define the composition operation, denoted $\cdot$, by $(f \cdot g)(q) = g(f(q))$, for all $q \in Q$. Then the set of all functions δ_w together with the composition operator form a monoid, with δ_ϵ as the identity element. This is called the *transition monoid* of $\mathcal{A}$. Instead of $(f \cdot g)$, we will simply write fg.

For a regular language $L \subseteq \Sigma^*$, if $\mathcal{A}$ is the minimal DFA such that $L = L(\mathcal{A})$, then as we saw in Theorem 4.2, based on the set of all transformations of the states of $\mathcal{A}$, we can construct an automaton $\mathcal{A}'$ to recognize $\mathrm{root}(L)$.

Though this new DFA, $\mathcal{A}'$, has all transformations of Q as its states, it is easy to see that the only reachable states are those that are a composition of the transformations $\delta_{a_1}, \ldots, \delta_{a_m}$, where $a_1, \ldots, a_m \in \Sigma$. The elements $\delta_{a_1}, \ldots, \delta_{a_m}$, and all of their compositions is precisely the transition monoid of $\mathcal{A}$. This fact gives us the following corollary.

Corollary 4.6. *For a regular language L, let $\mathcal{A}$ be the smallest DFA recognizing L. Then if M is the transition monoid of $\mathcal{A}$, we have that* $\mathrm{sc}(\mathrm{root}(L)) \leq |M|$.

Proof. In Theorem 4.2, the only reachable states in the construction of $\mathcal{A}'$ are those that belong to the transition monoid of $\mathcal{A}$. □

This connection between the state complexity of $\mathrm{root}(L)$ and the transition monoid of DFA for L allows us to exploit the results of Theorem 3.28 and Theorem 3.39. The bound

given in Corollary 4.6, however, is not necessarily tight. Even without the unreachable states, the DFA of Theorem 4.2 is not necessarily minimal. In the following section, we will show how to create a language based on a particular transition monoid, such that the number of equivalent states in the DFA of Theorem 4.2 is minimal.

4.2.1 Constructing Automata from Monoids

By associating an alphabet with the generators of a monoid, we can define a transition function for a DFA. The definition of the DFA is then completed by choosing a start state and a set of final states. This construction is given more formally below.

Let n, m be integers with $n, m \geq 1$. For a set of transformations $X = \{\alpha_1, \ldots, \alpha_m\}$, let $M \subseteq T_n$ denote the monoid generated by X. Then a *DFA based on* X is a DFA $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, F)$, where $|\Sigma| \geq m$, $z_0 \in Z_n$, $F \subseteq Z_n$, and δ is given by

$$\delta_a = \Psi(a), \text{ for all } a \in \Sigma,$$

for some map $\Psi : \Sigma \to X \cup \{\delta_\epsilon\}$ that is surjective on X.

Proposition 4.7. *Let $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, F)$ be a DFA. Then M is the transition monoid of $\mathcal{M}$ if and only if $\mathcal{M}$ is based on X, for some $X \subseteq T_n$ that generates the monoid M.*

Proof. For a DFA $\mathcal{M}$ based on X, the fact that M is the transition monoid of $\mathcal{M}$ is immediate from the construction. For any DFA $\mathcal{M}$ that has M as its transition monoid, we have that the set $\{\delta_a \in T_n : a \in \Sigma\}$ generates M. Then we can simply take Ψ given by $\Psi(a) = \delta_a$, for all $a \in \Sigma$. □

In particular, let $\mathcal{A}_{\Psi,X} = (Z_n, \Sigma, \delta, z_0, F)$ denote the DFA based on X when $z_0 = 1$, $F = \{1\}$, and Ψ is bijective on an m-element subset of Σ, with all other elements of Σ mapped to δ_ϵ. If Ψ_1 and Ψ_2 are maps over the same domain, then $\mathcal{A}_{\Psi_1,X}$ is isomorphic to $\mathcal{A}_{\Psi_2,X}$, up to a renaming of the states and alphabet symbols. For this reason, we will often denote this DFA simply by $\mathcal{A}_{\Sigma,X}$.

Example 4.8. Let $Y = \{\alpha, \beta\}$, where

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 5 & 3 \end{pmatrix}, \text{ and } \beta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 1 & 2 \end{pmatrix}.$$

Define Φ by $\Phi(a) = \alpha$ and $\Phi(b) = \beta$. Then Figure 4.9 depicts the DFA $\mathcal{A}_{\Phi,Y}$.

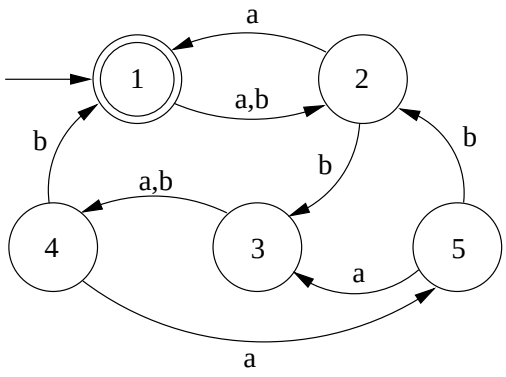

Figure 4.9: The automaton $\mathcal{A}_{\Phi,Y}$.

For $n \geq 5$, define $X_n \subseteq T_n$ to be a subset of transformations, and let M_n denote the monoid generated by X_n. In particular, define $X_{k,l} = \{\alpha, \beta\}$, where α and β are as in Lemma 3.30. Then $X_{k,l}$ generates $U_{k,l}$. We now state our main result concerning the state complexity of $\mathrm{root}(L(\mathcal{A}_{\Sigma,X_n}))$.

Theorem 4.10. *If $U_{k,l} \subseteq M_n$, for some coprime integers $k \geq 2$ and $l \geq 3$, with $k + l = n$, then the minimal DFA recognizing* $\mathrm{root}(L(\mathcal{A}_{\Sigma,X_n}))$ *has $|M_n| - \binom{n}{2}$ states.*

Before we are ready to prove this theorem, we must state a few more definitions and lemmas.

Definition. Let $\rho \in T_n$. For any i, j, k, if $\rho(i) = k = \rho(j)$ implies that $i = j$, then we say that k is *unique*.

Definition. Let $\rho \in T_n$ have rank 2, with $\mathrm{img}(\rho) = \{i, j\}$. Then by the *complement* of ρ, we mean the transformation $\overline{\rho} \in T_n$, where

$$\overline{\rho}(k) = \begin{cases} i, & \text{if } \rho(k) = j; \\ j, & \text{if } \rho(k) = i. \end{cases}$$

For example, if $\rho = \begin{pmatrix} 1 & 2 & 3 & \cdots & n-1 & n \\ 3 & 3 & 2 & \cdots & 2 & 2 \end{pmatrix}$, then $\overline{\rho} = \begin{pmatrix} 1 & 2 & 3 & \cdots & n-1 & n \\ 2 & 2 & 3 & \cdots & 3 & 3 \end{pmatrix}$.

It is easy to see that, in general, ρ and $\overline{\rho}$ have the same rank, and $\overline{\overline{\rho}} = \rho$.

For an automaton $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, F)$, define the DFA $\mathcal{M}^* = (M_n, \Sigma, \delta', \delta_\epsilon, F')$, where δ_ϵ is the identity element of T_n, $\delta'(\eta, a) = \eta\delta_a$ for all $\eta \in M_n$, for all $a \in \Sigma$, and $F' = \{\eta \in T_n : \eta(z_0) = z_0\}$. Then $L(\mathcal{M}^*) = \text{root}(L(\mathcal{M}))$.

For $\eta, \theta \in M_n$, with $\eta \neq \theta$, note that η and θ are equivalent states if and only if for all $\rho \in \Sigma^*$ we have that

$$\delta'(\eta, \rho) \in F' \Leftrightarrow \delta'(\theta, \rho) \in F'.$$

However, since $\delta'(\eta, \rho) = \eta\delta_\rho$, this is equivalent to saying that η and θ are equivalent states in M if and only if for all $\rho \in M_n$, we have

$$\eta\rho \in F' \Leftrightarrow \theta\rho \in F'.$$

Lemma 4.11. *Let $Y \subseteq T_n$ generate M_n, and let $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, F)$ be an automaton based on Y such that $z_0 \in F$. Let $\eta, \theta \in M_n$, with $\eta \neq \theta$ and $\text{rank}(\eta) = 2$. If $\eta(z_0)$ is unique in the image of η, and $\overline{\eta} = \theta$, then η and θ are equivalent states in $\mathcal{M}^*$.*

Proof. We have

$$\eta = \begin{pmatrix} 1 & 2 & \cdots & z_0 - 1 & z_0 & z_0 + 1 & \cdots & n \\ j & j & \cdots & j & i & j & \cdots & j \end{pmatrix},$$

and

$$\theta = \begin{pmatrix} 1 & 2 & \cdots & z_0 - 1 & z_0 & z_0 + 1 & \cdots & n \\ i & i & \cdots & i & j & i & \cdots & i \end{pmatrix},$$

for some $i \neq j$.

If $\eta(z_0) \in F$, then $\eta \in F'$. If $\theta(z_0) \in F$, then $\theta \in F'$. Otherwise, $\theta(z_0) \in Z_n \backslash \{z_0\}$. Since $\theta(z) = \eta(z_0) \in F$ for all $z \in Z_n \backslash \{z_0\}$, we have that $\theta^2(z_0) \in F$, and hence $\theta \in F'$. Similarly, $\theta(z_0) \in F$ implies that $\eta, \theta \in F'$. Furthermore, if $\text{img}(\eta) \cap F = \emptyset$, then $\text{img}(\eta^n) \cap F = \emptyset$, for all n, and hence $\eta^n(z_0) \notin F$ so that $\eta \notin F'$. Since $\text{img}(\eta) = \text{img}(\theta)$, this gives $\theta \notin F'$. Therefore $\eta \in F'$ if and only if $\theta \in F'$.

Let $\rho \in M_n$. Since η and θ have rank 2, $\eta\rho$ and $\theta\rho$ have rank ≤ 2. If $\eta\rho$ has rank 2, then $\rho(i) \neq \rho(j)$, so that $\overline{\eta\rho} = \theta\rho$. Hence $\eta\rho \in F'$ if and only if $\theta\rho \in F'$. The argument is the same for the case where $\theta\rho$ has rank 2. Now, if $\eta\rho$ has rank 1, then we must have $\rho = \rho'\sigma\rho''$, where $\sigma(s) = \sigma(t)$ for some s and t, and where ρ' is a permutation such that either $\eta\rho'(z_0) = s$ and $\eta\rho'(z) = t$, for all $z \neq z_0$, or $\eta\rho'(z_0) = t$ and $\eta\rho'(z) = s$, for all $z \neq z_0$. Without loss of generality, assume the former. Then clearly $\theta\rho'(z_0) = t$ and $\theta\rho'(z) = s$, for all $z \neq z_0$. It follows that $\eta\rho'\sigma = \theta\rho'\sigma$, so that $\eta\rho = \theta\rho$.

Therefore η and θ are equivalent states. □

Lemma 4.12. *Let $Y \subseteq T_n$ generate M_n, and let $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, F)$ be an automaton based on Y, such that $z_0 \notin F$. Let η, $\theta \in M_n$, with $\eta(z_0)$ unique in the image of η,* $\mathrm{rank}(\eta) = 2$, *and* $\mathrm{img}(\eta) = \mathrm{img}(\theta)$. *If $\theta(z_0) = \eta(z_0)$, and if $\theta(z) = \eta(z_0)$ implies that $z \in F$, then η and θ are equivalent states in $\mathcal{M}^*$.*

Proof. If $\eta(z_0) \in F$, then $\eta \in F'$. Since $\theta(z_0) = \eta(z_0)$, it follows that $\theta \in F'$. Now suppose that $\eta(z_0) \notin F$. If $\eta \in F'$, then since $\mathrm{rank}(\eta) = 2$, we must have that $\eta^2(z_0) \in F$. Now $\theta(z_0) \notin F$, so $\theta(\theta(z_0)) \neq \eta(z_0)$. But since $\mathrm{rank}(\theta) = 2$, and $\mathrm{img}(\theta) = \mathrm{img}(\eta)$, it follows that $\theta^2(z_0) = \eta^2(z_0)$. Hence $\theta \in F'$. If $\eta \notin F'$, then $\eta(z_0) = z_0$ or $\mathrm{img}(\eta) \cap F = \emptyset$. In either case, this implies that $\theta \notin F'$. Therefore $\eta \in F'$ if and only if $\theta \in F'$.

Let $\rho \in M_n$. Then, following an argument similar to the one used in the proof of Lemma 4.11, we have that $\eta\rho \in F'$ if and only if $\theta\rho \in F'$. Therefore η and θ are equivalent states. □

Now that we have a characterization of equivalent states in the general case for $\mathcal{M}^*$, we turn our attention toward the specific case, for $\mathcal{A}^*_{\Sigma,X_n}$.

Lemma 4.13. *Let η, $\theta \in M_n$, with $\eta \neq \theta$. If* $\mathrm{rank}(\eta) = 1$, *then η and θ are not equivalent states in $\mathcal{A}^*_{\Sigma,X_n}$.*

Proof. Since η has rank 1, we have that $\mathrm{img}(\eta) = \{z_1\}$ for some z_1. If $\eta(1) \neq \theta(1)$, then take $\rho \in U_{k,l}$ such that $\rho(z_1) = 2$, and $\rho(z) = 1$, for all $z \neq z_1$. Then $\mathrm{img}(\eta\rho) = \{2\}$, so that $\eta\rho \notin F'$. But $\theta\rho(1) = 1$, so that $\theta\rho \in F'$. Hence η and θ are not equivalent. If $\eta(1) = \theta(1)$, then $\mathrm{rank}(\theta) \neq 1$ so that for some $z_2 \neq 1$ we have $\theta(z_2) \neq z_1$. Take $\rho \in U_{k,l}$ such that $\rho(\theta(z_2)) = 1$, and $\rho(z) = z_2$, for all $z \neq \theta(z_2)$. Then $\mathrm{img}(\eta\rho) = \{z_2\}$, so that $\eta\rho \notin F'$. But $(\theta\rho)^2(1) = 1$, so that $\theta\rho \in F'$. Hence η and θ are not equivalent. □

Lemma 4.14. *For η, $\theta \in M_n$, with $\eta \neq \theta$, let η have rank 2. Then η and θ are equivalent states in $\mathcal{A}^*_{\Sigma,X_n}$ if and only if $\eta(1)$ is unique in the image of η, and $\overline{\eta} = \theta$.*

Proof. Let $\mathrm{img}(\eta) = \{i, j\}$ for some i, j. Without loss of generality, assume that $\eta(1) = i$.

Since $1 \in F$, Lemma 4.11 applies and gives the result in the forward direction. For the other direction, we have two cases.

Case 1. i is not unique in the image of η.

Case 1.a. $i = \eta(1) \neq \theta(1)$.

Since i is not unique, $\eta(z) = i$, for some $z \neq 1$. Choose $\rho \in U_{k,l}$ such that $\rho(i) = z$, and $\rho(\theta(1)) = 1$. Then $\eta\rho(1) = \eta\rho(z) = z$ so that $(\eta\rho)^n(1) = z$, for all $n \geq 0$. This gives us $\eta\rho \notin F'$. But $\theta\rho(1) = 1$, so that $\theta\rho \in F'$.

Case 1.b. $i = \eta(1) = \theta(1)$.

Case 1.b.i. $\mathrm{img}(\eta) = \mathrm{img}(\theta)$.

Since $\mathrm{img}(\eta) = \mathrm{img}(\theta)$ and $\eta \neq \theta$, then for some $z \neq 1$, we must have either $i = \eta(z) \neq \theta(z) = j$, or $j = \eta(z) \neq \theta(z) = i$. Without loss of generality, assume the former. Choose $\rho \in U_{k,l}$ such that $\rho(i) = z$, and $\rho(j) = 1$. Then $(\eta\rho)^n(1) = z$, for all $n \geq 0$, and $(\theta\rho)^2(1) = 1$. This gives us $\eta\rho \notin F'$ and $\theta\rho \in F'$.

Case 1.b.ii. $\mathrm{img}(\eta) \neq \mathrm{img}(\theta)$.

If $\mathrm{rank}(\theta) = 1$ then by Lemma 4.13, η and θ are not equivalent. Otherwise there exists some $z_1 \in Z_n$ such that $\theta(z_1) \notin \mathrm{img}(\eta)$. Choose $\rho \in U_{k,l}$ such that $\rho(\theta(z_1)) = 1$, and $\rho(z) = 2$, for all $z \neq \theta(z_1)$. Then $\eta\rho \neq \theta\rho$ and $\mathrm{rank}(\eta\rho) = 1$, and so by Lemma 4.13, $\eta\rho$ and $\theta\rho$ are not equivalent. Hence η and θ are not equivalent.

Case 2. i is unique in the image of η, and $\overline{\eta} \neq \theta$.

If $\text{rank}(\theta) = 1$ then by Lemma 4.13, η and θ are not equivalent. If $\text{img}(\eta) = \text{img}(\theta)$, then since $\overline{\eta} \neq \theta$, we have that $\theta(1)$ is not unique. So we can reverse the roles of η and θ and apply case 1 to get the desired result. Assume then, that $\text{img}(\eta) \neq \text{img}(\theta)$, and $\text{rank}(\theta) \geq 2$. Then the fact that η and θ are not equivalent follows just as in case 1.b.ii.

Therefore, η, and θ are equivalent states if and only if $\eta(1)$ is unique in the image of η, and $\overline{\eta} = \theta$. □

Lemma 4.15. *Let η, $\theta \in M_n$, with $\eta \neq \theta$. If η, θ have rank ≥ 3, then η and θ are not equivalent states in $\mathcal{A}^*_{\Sigma,X_n}$.*

Proof. Since $\eta \neq \theta$, there exists some $z_1 \in Z_n$ such that $\eta(z_1) \neq \theta(z_1)$. Let $z_2 = \eta(z_1)$. Take $\rho \in U_{k,l}$ such that $\rho(z_2) = 1$, and $\rho(z) = 2$, for all $z \neq z_2$. Since $\text{rank}(\eta) \geq 3$, we have $\text{rank}(\eta\rho) = 2$. If $\eta\rho(1)$ is not unique, then by Lemma 4.14, $\eta\rho$ and $\theta\rho$ are not equivalent. Hence η and θ are not equivalent. If $\eta\rho(1)$ is unique, then it must be that $z_1 = 1$. Furthermore, since $\text{rank}(\theta) \geq 3$, we cannot have $\theta(z) = z_2$ for all $z \neq 1$, so we cannot have $\theta\rho(z) = 1$ for all $z \neq 1$. Therefore $\theta\rho \neq \overline{\eta\rho}$. Then by Lemma 4.14, $\eta\rho$ and $\theta\rho$ are not equivalent. Hence η and θ are not equivalent. □

We are now ready to prove Theorem 4.10.

Proof (Theorem 4.10). Lemma 4.13 – 4.15 cover all possible cases for $\eta, \theta \in M_n$, $\eta \neq \theta$. Therefore, two states are equivalent if and only if they satisfy the hypothesis of Lemma 4.11. There are $\binom{n}{2}$ such equivalence classes in $U_{k,l} \subseteq M_n$, each containing exactly 2 elements. All other elements of M_n are in equivalence classes by themselves. It follows that the minimal DFA recognizing $\text{root}(L(\mathcal{A}_{\Sigma,X_n}))$ has $|M_n| - \binom{n}{2}$ states. □

4.2.2 Alphabets of Size Two

Now that we have established a close relationship between $\text{sc}(\text{root}(L))$ and the transition monoid of the the minimal automaton recognizing L, we can take advantage of results

concerning the size of the largest monoids to give bounds on the worst-case state complexity of root(L).

An immediate result of Corollary 3.19 gives us the following upper bound.

Corollary 4.16. *For any binary language L, we have*

$$\mathrm{sc}(\mathrm{root}(L)) \leq n^n - n! + g(n),$$

where $g(n)$ is Landau's function.

The following corollary to Theorem 4.10 gives a lower bound for alphabets of size two. It also proves the existence of a sequence of regular binary languages with state complexity n whose root has a state complexity that approaches n^n as n increases without bound.

We now state our first main result of this chapter.

Corollary 4.17. *For $n \geq 7$, there exists a regular language L over an alphabet of size 2, with $\mathrm{sc}(L) \leq n$, such that*

$$\mathrm{sc}(\mathrm{root}(L)) \geq n^n \left(1 - \sqrt{2}\left(\frac{2}{e}\right)^{\frac{n}{2}} e^{\frac{1}{12}} - \sqrt{8}\frac{1}{\sqrt{n}} e^{\frac{1}{12}}\right) - \binom{n}{2}.$$

Proof. The result follows from a combination of Theorem 3.32 and Theorem 4.10. □

In contrast to the operations summarized in Table 1.2, whose state complexity is $O(2^n)$, Corollary 4.17 gives us a relatively simple operation on regular languages whose worst-case state complexity is $n^n(1 - O(n^{-\frac{1}{2}}))$, even for binary alphabets.

Theorem 4.10 does not apply when $l = 2$, but unfortunately, Theorem 3.39 does not exclude this possibility. To guarantee that this fact is of no consequence, we must show that not only is the monoid $U_{n-2,2}$ never the largest, but that it is also smaller than the largest monoid by at least $\binom{n}{2}$ elements. The following lemma demonstrates this.

Lemma 4.18. *For $n \geq 7$, we have that*

$$|U_{2,n-2}| - |U_{n-2,2}| \geq \binom{n}{2}.$$

Proof (Lemma 4.18). As stated in [10], for $k+l=n$, we have the following formula

$$|U_{k,l}| = kl + \sum_{i=1}^{n}\left(\binom{n}{i} - \binom{k}{i-l}\right)\left(\left\{{n \atop i}\right\} - \sum_{r=1}^{i}\left\{{k \atop r}\right\}\left\{{l \atop i-r}\right\}\right)i!,$$

where $\left\{{n \atop i}\right\}$ is a Stirling number of the second kind, the number of ways to partition a set of n elements into i non-empty sets. This gives

$$|U_{k,l}| - |U_{l,k}| = \sum_{i=1}^{n}\left(\binom{l}{i-k} - \binom{k}{i-l}\right)\left(\left\{{n \atop i}\right\} - \sum_{r=1}^{i}\left\{{k \atop r}\right\}\left\{{l \atop i-r}\right\}\right)i!. \qquad (*)$$

Since $\left\{{n \atop k}\right\} = 0$ whenever $k > n$ or $k < 1$, for $k = 2$, and $l = n-2$, we have

$$\sum_{r=1}^{i}\left\{{2 \atop r}\right\}\left\{{n-2 \atop i-r}\right\} = \left\{{n-2 \atop i-2}\right\} + \left\{{n-2 \atop i-1}\right\}.$$

Also, notice that $\binom{n-2}{i-2} - \binom{2}{i-n+2}$ is positive when $2 \geq i \geq n-1$, and zero otherwise, so that $(*)$ becomes

$$|U_{2,n-2}| - |U_{n-2,2}| \geq \sum_{i=2}^{n-1}\left(\left\{{n \atop i}\right\} - \left\{{n-2 \atop i-2}\right\} - \left\{{n-2 \atop i-1}\right\}\right)i!.$$

And finally, using the identity $\left\{{n \atop k}\right\} = \left\{{n-1 \atop k-1}\right\} + k\left\{{n-1 \atop k}\right\}$, we see that

$$\left\{{n \atop i}\right\} = \left\{{n-2 \atop i-2}\right\} + (2i-1)\left\{{n-2 \atop i-1}\right\} + (i-1)\left\{{n-2 \atop i}\right\},$$

so that we get

$$|U_{2,n-2}| - |U_{n-2,2}| \geq \sum_{i=2}^{n-1}\left((2i-2)\left\{{n-2 \atop i-1}\right\} + (i-1)\left\{{n-2 \atop i}\right\}\right)i! \geq \sum_{i=2}^{n-1} i! \geq \binom{n}{2}.$$

□

The choice of start and final states in the construction of the DFA $\mathcal{A}_{\Sigma,X_n}$ is the best possible. The following theorem will show that for any other DFA with the same transition function, a different assignment of start and final states will not increase the state complexity of the language it recognizes.

Theorem 4.19. *Let $Y \subseteq T_n$ generate M_n, and let $\mathcal{M} = (Z_n, \Sigma, \delta, z_0, G)$ be an automaton based on Y. Then* $\mathrm{sc}(\mathrm{root}(L(\mathcal{M}))) \leq \mathrm{sc}(\mathrm{root}(L(\mathcal{A}_{\Sigma,X_n}))$.

Proof. If $z_0 \in G$, then Lemma 4.11 applies. It follows that there are at least $\binom{n}{2}$ pairs of equivalent states in M^*. If $z_0 \notin G$, then Lemma 4.12 applies, and again we have at least $\binom{n}{2}$ pairs of equivalent states in M^*. In either case, this gives

$$\mathrm{sc}(\mathrm{root}(L(\mathcal{M}))) \leq |M_n| - \binom{n}{2} \leq \mathrm{sc}(\mathrm{root}(L(\mathcal{A}_{\Sigma,X_n})).$$

□

We now state our second main result of this chapter.

Corollary 4.20. *For prime numbers $n \geq 7$, there exist positive, coprime integers $k \geq 2$, $l \geq 3$, with $k+l = n$, such that if L is a language over an alphabet of size 2, with* $\mathrm{sc}(L) \leq n$, *then* $\mathrm{sc}(\mathrm{root}(L)) \leq |U_{k,l}| - \binom{n}{2}$. *Furthermore, this bound is tight.*

Proof. Let U' denote the largest two-generated submonoid of T_n. Then by Theorem 3.39 and Lemma 4.18, we have that $U' = U_{k',l'}$ for some coprime integers $k' \geq 2$, $l' \geq 3$ with $k' + l' = n$.

Let $\mathcal{M}$ be the smallest DFA recognizing L, and let M be the transition monoid of $\mathcal{M}$. If M is of the form $U_{k,l}$, with $k \geq 2$, $l \geq 3$, then $|U_{k,l}| \leq |U'|$. It follows from Theorem 4.19 that $\mathrm{sc}(\mathrm{root}(L)) \leq |U_{k,l}| - \binom{n}{2} \leq |U'| - \binom{n}{2}$. If M is of the form $U_{k,l}$, with $k = n-2$, $l = 2$, then by Corollary 4.6 and Lemma 4.18 we have $\mathrm{sc}(\mathrm{root}(L)) \leq |U_{n-2,2}| \leq |U_{2,n-2}| - \binom{n}{2} \leq |U'| - \binom{n}{2}$.

Let V denote the largest two-generated submonoid of T_n that is not of the form $U_{k,l}$ for some coprime integers $k \geq 2$, $l \geq 3$ with $k + l = n$. Again, by Theorem 3.39 we have that $|U'| - |V| \geq \binom{n}{2}$. It follows from Corollary 4.6 that if M is not of the form $U_{k,l}$, we have

$$\mathrm{sc}(\mathrm{root}(L)) \leq |M| \leq |V| \leq |U_{k,l}| - \binom{n}{2}.$$

The fact that the bound is tight is an immediate consequence of Theorem 4.10. □

If Conjecture 3.42 is true, then the result stated in the following conjecture is an immediate result.

Conjecture 4.21. *For and integer $n \geq 7$, there exist positive, coprime integers $k \geq 2$, $l \geq 3$, with $k+l = n$, such that if L is a language over an alphabet of size 2, with $\mathrm{sc}(L) \leq n$, then $\mathrm{sc}(\mathrm{root}(L)) \leq |U_{k,l}| - \binom{n}{2}$. This bound is tight.*

4.2.3 Alphabets of Size Three or More

The results concerning the largest monoid on ≥ 3 generators are definite and much simpler. For this reason, on alphabets of size ≥ 3 we are able to give a much better bound.

We now state our third main result of this chapter.

Theorem 4.22. *Let Σ be an alphabet of size $m \geq 3$. For $n \geq 1$, if L is a language over Σ with $\mathrm{sc}(L) \leq n$, then $\mathrm{sc}(\mathrm{root}(L)) \leq n^n - \binom{n}{2}$. Furthermore, this bound is tight.*

Proof. Define M to be the transition monoid of the smallest DFA recognizing L. If $|M| \leq n^n - \binom{n}{2}$, then certainly $\mathrm{sc}(\mathrm{root}(L)) \leq n^n - \binom{n}{2}$. So suppose that $|M| > n^n - \binom{n}{2}$. Then it follows from Lemma 3.7 that $M = T_n$.

For $1 \leq n \leq 6$, this author verified computationally that if the transition monoid of the minimal DFA recognizing L is T_n, then $\mathrm{sc}(\mathrm{root}(L)) = n^n - \binom{n}{2}$. For $n \geq 7$, if the transition monoid is T_n, then clearly $U_{k,l} \subseteq T_n$ for some suitable k, l so that Theorem 4.10 applies, and hence $\mathrm{sc}(\mathrm{root}(L)) = n^n - \binom{n}{2}$.

To show that the bound is tight, it suffices to show that for any n there exists a language L over Σ such that the transition monoid of the minimal DFA recognizing L is T_n. Let X be a set of transformations such that $|X| = \min(n, 3)$ and X generates T_n. For $n \in \{1, 2\}$, the fact that such an X exists is easy to check. For $n \geq 3$, the existence of X follows from Lemma 3.6. Then the language $L(\mathcal{A}_{\Sigma,X})$, gives the desired result. □

4.3 An Application to the State Complexity of 2DFAs

As noted in the introduction to this chapter, the n^n states used by the DFA of Theorem 4.2, and the $2n$ states used by the 2DFA of Theorem 4.1 suggest that there is at least an $n^{O(n)}$ blow-up in the number of states when converting from a 2DFA to a DFA. This bound was proven by Shepherdson [33], who showed that given an n-state 2DFA it was possible to construct an $n(n+1)^n$ state DFA to accept the same language. Birget [4] improved

Shepherdson's result by showing that an equivalent DFA can be constructed using only n^n states. Unfortunately, these general constructions give no indication that the bound is tight. To date, the best known results show that the worst-case lower bound is $n^{\Theta(n)}$. Two examples are presented here.

Let $\Sigma = a, b, c$, and define the DFA $\mathcal{B}_m = (Z_m, \Sigma, \delta, 1, m)$, where δ is given by $\delta_a = (2\ 3\ \cdots\ m\ 1)$, $\delta_b = (1\ 2)$, and

$$\delta_c = \begin{pmatrix} 1 & 2 & 3 & \cdots & k \\ 1 & 1 & 3 & \cdots & k \end{pmatrix}.$$

So, δ_a is a m-cycle permutation, δ_b is a transposition, and δ_c is a transformation of rank $m-1$. Then $w \in L(\mathcal{B}_m)$ if and only if $w(1) = m$ when w is viewed as a transformation of Z_n.

Now define the language $F_m = \{xdy : x, y \in \Sigma^* \text{ and } yx \in L(\mathcal{B}_m)\}$. So F_m is a language over the alphabet $\{a, b, c, d\}$. Moore [20] showed the language F_m was recognizable by a 2DFA with end-markers using $2m+5$ states, but that the minimal DFA recognizing F_m had exactly $m^m + m(2^m - 2) + 2$ states. Then with $n = 2m+5$, it follows that the function

$$f(n) = \left(\frac{n-5}{2}\right)^{\frac{n-5}{2}} + \left(\frac{n-5}{2}\right)\left(2^{\frac{n-5}{2}} - 2\right) + 2,$$

is a lower bound on the worst-case blow-up of an n-state 2DFA. Furthermore, $f(n) \in n^{\Theta(n)}$.

The second example of an $n^{\Theta(n)}$ blow-up is based on the language G_m, defined by

$$G_m = \{0^{i_1}10^{i_2}1\cdots10^{i_m}2^k0^{i_k} : 1 \le k \le m, \text{ and } 1 \le i_j \le m, \text{ for } j \in \{1, \ldots, m\}\}.$$

So G_m is a language over the alphabet $\{0, 1, 2\}$. Meyer and Fischer [18] claimed that the language G_m can be recognized by a 2DFA with end-markers using $5m+5$ states. Moreover, they claimed that the smallest DFA recognizing G_m has at least m^m states. Then for $n = 5m+5$, this gives a lower bound of $n^{\Theta(n)}$ on the worst-case blow-up. To get an idea of the upper bound of this example, consider the following lemma, which gives an upper bound on the size of the minimal DFA for G_m.

Lemma 4.23. *For sufficiently large m, we have that the minimal DFA recognizing the language G_m has at most $2 \cdot m^{m+2}$ states.*

Proof. The strings in G_m consist of m blocks of 0's separated singleton 1's. Near the end of each string is a block of 2's, the length of which indicates which block of 0's should be repeated at the end.

Now, with respect to the relation $\sim_{G_m}$, consider the equivalence classes of the strings in Σ^*. Certainly ϵ is in an equivalence class by itself since it is the only string $x \in \Sigma^*$ for which $xy \in G_m$ for all $y \in G_m$.

For a string x of the form 0^i, $1 \leq i \leq m$. Let $y \in \Sigma^*$ be some prefix of a word in G_m. If y is of the same form as x, but not equal, then $y = 0^j$, $1 \leq j \leq m$, and $j \neq i$. Then there exists a string with a suitable number of leading 0's that will distinguish x and y. If y is not of the same form as x, then y must contain a 1. Then there exists a string with $m-1$ 1's that will distinguish x and y. Hence x is in an equivalence class by itself. This gives m classes.

In general, it is not hard to see that if x is of the form

$$0^{i_1}10^{i_2}1\cdots10^{i_s}, \quad 1 \leq s \leq m,\ 1 \leq i_j \leq m \text{ for } j \in \{1,\ldots,s\},$$

then x is in an equivalence class by itself. To count these classes, consider the number of strings of this form when s is fixed. Each string has s blocks of 0's, and up to m 0's in each block. This gives m^s strings. Then in total we have $\sum_{1\leq s\leq m} m^s = \frac{m^{m+1}-m}{m-1}$ classes.

Furthermore, it is easy to see that x is in an equivalence class by itself when x is in one of the following forms:

- $0^{i_1}10^{i_2}1\cdots10^{i_r}1, \quad 1 \leq r \leq m-1,\ 1 \leq i_j \leq m \text{ for } j \in \{1,\ldots,r\}$;
- $0^{i_1}10^{i_2}1\cdots10^{i_m}2, \quad 1 \leq i_j \leq m \text{ for } j \in \{1,\ldots,m\}$.

Then by similar counting we get $\frac{m^{m+1}-m}{m-1}$ equivalence classes. Now, strings of the form

$$0^{i_1}10^{i_2}1\cdots10^{i_m}2^k, \quad 2 \leq k \leq m,\ 1 \leq i_j \leq m \text{ for } j \in \{1,\ldots,m\},$$

can account for no more than m^{m+1} classes. And strings of the form

$$0^{i_1}10^{i_2}1\cdots10^{i_m}2^k0^r, \quad 2 \leq k \leq m,\ 1 \leq r \leq m,\ 1 \leq i_j \leq m \text{ for } j \in \{1,\ldots,m\},$$

can account for no more than m^{m+2} classes. All other strings not yet discussed are contained in a single class, since they are not prefixes of any word in G_m.

In total, this gives us at most $2 \cdot \frac{m^{m+1}-m}{m-1} + m^{m+1} + m^{m+2} + 2$ classes. For sufficiently large m, this quantity is less than $2 \cdot m^{m+2}$, as required. □

If $n = 5m + 5$, then it follows from Lemma 4.23 that the n-state 2DFA construction given by Meyer and Fischer for the language G_m has a blow-up of at most

$$g(n) = 2 \cdot \left(\frac{n-5}{5}\right)^{\frac{n}{5}+1},$$

states when converted to a DFA.

As it turns out, our results for the state complexity of root(L) can be used to improve the bounds demonstrated for the languages F_m and G_m. Recall the definition of the automaton $\mathcal{A}_{\Sigma,X_{k,l}}$ from Section 4.2.1. For all $n \geq 7$, let k, l be coprime integers such that the bound of Theorem 3.28 is achieved. Then define $H_n = \text{root}(L(\mathcal{A}_{\Sigma,X_{k,l}}))$.

Theorem 4.24. *For sufficiently large n, there exists an n-state 2DFA with end-markers such that the equivalent DFA has at least*

$$h(n) = \left(\frac{n}{2}\right)^{\frac{n}{2}} - 4 \cdot \left(\frac{n}{2}\right)^{\frac{n}{2}-1} - \frac{1}{8}n^2,$$

states.

Proof. Let $n = 2m$. Then by Theorem 4.1, the language H_m is recognized by a 2DFA with n states. It follows from Theorem 3.28 and Theorem 4.10 that for sufficiently large m we have

$$\begin{aligned}
\text{sc}(H_m) &\geq m^m - 4m^{m-1} - \binom{m}{2} \\
&\geq m^m - 4m^{m-1} - \frac{1}{2}m^2 \\
&\geq \left(\frac{n}{2}\right)^{\frac{n}{2}} - 4 \cdot \left(\frac{n}{2}\right)^{\frac{n}{2}-1} - \frac{1}{8}n^2,
\end{aligned}$$

as required. □

Hence, Theorem 4.24 gives us yet another example of an $n^{\Theta(n)}$ blow-up in the number of states when converting from a 2DFA to a DFA. Furthermore, when we consider that the

ratio

$$\frac{h(n)}{f(n)} \in \Theta\left(n^{\frac{5}{2}}\right),$$

and the ratio

$$\frac{h(n)}{g(n)} \in \Theta\left(\left(\frac{25}{32}\right)^{\frac{1}{10}n} n^{\frac{3}{10}n}\right),$$

we see that Theorem 4.24 actually gives a significant improvement over the results due to Moore [20] and Meyer and Fischer [18].

Despite this improvement of the worst-case lower bound, Corollary 4.3 shows that the worst case achievable by a 2DFA recognizing a language of the form root(L) is less than $n^{n/2}$. It is clear then that we cannot use root(L) to prove that the upper bound of n^n on 2DFA-to-DFA conversion is tight. Hence, the problem of finding a tight bound remains open (see Open Problem 5.2).

Chapter 5

Open Problems

In this chapter, we state some open problems, given in the context of the results presented in this thesis.

In Section 2.2, Theorem 2.4, proved by Ellul, Krawetz, Shallit, and Wang [8], gives us an improved lower bound of $\frac{5}{4} \cdot 2^{n/2}$ on the worst case state complexity of a regular expression. However, with an upper bound of $2^n + 1$, there is still a significant gap.

Open Problem 5.1. *Give a tight upper bound on the state complexity of a regular expression.*

In Section 4.3 we returned to the problem of determining the state complexity of other representations of regular languages. In particular, Theorem 4.24 improved on the results of Moore [20], and Meyer and Fischer [18] concerning the lower bound of the worst-case state complexity of a language recognized by an n-state 2DFA. As noted, however, we were not able to show that the upper bound of n^n, proved by Birget [4], is tight. So we have the following problem.

Open Problem 5.2. *Let L be a language over an alphabet of size $k \geq 1$, and recognized by an n-state 2DFA with end-markers. Give a tight upper bound on the state complexity of L.*

The results of Chapter 3 are relevant to more than just theoretical computer science, though it has perhaps been our main motivation. The results are applicable to any finitely generated monoid.

It is clear that the problem of the maximal monoid on two generators is not solved with the proof of Corollary 3.40. In fact, even in the prime case, we are not certain how to choose the integers k, l so that $U_{k,l}$ is maximal.

Open Problem 5.3. *For n prime, determine the values of k, l such that the monoid $U_{k,l}$ is maximal.*

Open Problem 5.4. *For composite $n > 7$, determine the largest two-generated monoid of transformations of Z_n.*

Actually, Open Problem 5.4 reduces to determining the analog of Lemma 3.33, Lemma 3.34, and Corollary 3.35, for values of n that are not prime. All other facts leading to the proof of Theorem 3.39 are proven for any n.

In Chapter 4, Corollary 4.17 gives us the existence of an infinite family of languages whose root has state complexity approaching n^n, for alphabets of size two or more. In Theorem 4.22 we obtain a tight upper bound on the state complexity of languages over arbitrarily sized alphabets. In the case of binary languages, Corollary 4.20 gives us a tight upper bound when state complexity of the original language is prime. Though this bound is not specific, any result concerning 5.3 will apply immediately. This leaves us with the case for binary languages with non-prime state complexity.

Open Problem 5.5. *For composite $n > 7$, if L is a binary language with state complexity n, what is the tight upper bound on the state complexity of* $\mathrm{root}(L)$*?*

Of course, in the case for n not prime, since we do not know the structure of the largest two-generated monoid, it is possible that results similar to Theorem 4.10 do not hold. However, given the size of the $U_{k,l}$ monoids, it seems unlikely that any monoid large enough would not contain the elements necessary for the techniques used in the proof of Theorem 4.10 to apply.

Open Problem 5.6. *For any monoid $M \subseteq T_n$, with $|M| \geq |U_{k,l}|$, for all k, l, if M is the transition monoid of a minimal DFA $\mathcal{A}$, is* $\mathrm{sc}(\mathrm{root}(L(\mathcal{A}))) = |M| - \binom{n}{2}$*?*

A solution to Open Problem 5.6 would guarantee that any results concerning the size of the largest two-generated monoid would apply to the state complexity of $\mathrm{root}(L)$. Currently, without having a proof of Conjecture 3.41, it is impossible to say whether or not we will be able to use such a result to prove Conjecture 4.21.

We conclude with a problem whose solution could prove useful in the pursuit of a solution to Open Problem 5.5. No effort was made to improve the size of the gap between $U_{k,l}$ and other more general monoids, proved in Lemma 3.34. We chose this bound because it served a specific purpose in the proof of Corollary 4.20. It is almost certain that this bound can be improved upon.

Open Problem 5.7. *Let V be the largest monoid generated by a set $\{\alpha, \beta\}$, where $\alpha \in S_n$ and $\beta \in T_n$ is a non-bijective transformation that identifies two elements of the same cycle in α. Improve the bound*

$$|U_{k,l}| - |V| \geq \binom{n}{2},$$

for some $n = k + l$.

Bibliography

[1] M. Artin. *Algebra*. Prentice-Hall, 1991.

[2] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume 1: Efficient Algorithms*. The MIT Press, 1996.

[3] J.-C. Birget. Intersection and union of regular languages and state complexity. *Inform. Process. Lett.* **43** (1992), 185–190.

[4] J.-C. Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Math. Systems Theory* **26** (1993), 237–269.

[5] J. Dénes. On transformations, transformation-semigroups and graphs. In P. Erdös and G. Katona, editors, *Theory of Graphs: Proc. Colloq. Graph Theory (Tihany 1966)*, pp. 65–75. Academic Press, 1968.

[6] J. Dénes. On a generalization of permutations: some properties of transformations. In *Permutations: Actes du Colloque sur Les Permutations (Paris 1972)*, pp. 117–120. Gauthier-Villars, 1972.

[7] K. Ellul. Descriptional complexity measures of regular languages. Master's thesis, University of Waterloo, 2002.

[8] K. Ellul, B. Krawetz, J. Shallit, and M. w. Wang. Regular expressions: new results and open problems. To appear in *J. Automata, Languages, and Combinatorics*.

[9] M. Holzer and B. König. On deterministic finite automata and syntactic monoid size. In M. Ito and M. Toyama, editors, *Proc. Developments in Language Theory 2002*, Vol. 2450 of *Lecture Notes in Computer Science*, pp. 229–240. Springer-Verlag, 2003.

[10] M. Holzer and B. König. On deterministic finite automata and syntactic monoid size, continued. In Z. Ésik and Z. Fülöp, editors, *Proc. Developments in Language Theory 2003*, Vol. 2710 of *Lecture Notes in Computer Science*, pp. 349–360. Springer-Verlag, 2003.

[11] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[12] S. Horváth, P. Leupold, and G. Lischke. Roots and powers of regular languages. In M. Ito and M. Toyama, editors, *Proc. Developments in Language Theory 2002*, Vol. 2450 of *Lecture Notes in Computer Science*, pp. 220–230. Springer-Verlag, 2003.

[13] B. Krawetz, J. Lawrence, and J. Shallit. State complexity and the monoid of transformations of a finite set. Preprint. Available at `http://arxiv.org/math/0306416v1`.

[14] G. Lallement. *Semigroups and Combinatorial Applications*. John Wiley and Sons, 1979.

[15] E. Landau. Über die Maximalordung der Permutation gegebenen Grades. *Archiv der Mathematik und Physik*, Ser. 3, **5** (1903), 92–103.

[16] F. Lazebnik. New upper bounds for the greatest number of proper colorings of a (v, e)-graph. *J. Graph Theory* **14** (1990), 25–29.

[17] H. Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.* **27** (1998), 1073–1082.

[18] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proc. 12th IEEE Symp. Switching and Automata Theory*, pp. 188–190, 1971.

[19] W. Miller. The maximum order of an element of a finite symmetric group. *Amer. Math. Monthly* **94** (1987), 497–506.

[20] F. R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata. *IEEE Trans. Comput.* **20** (1971), 1211–1214.

[21] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.* **9** (1958), 541–544.

[22] J.-L. Nicolas. On Landau's function $g(n)$. In R. L. Graham and J. Nesetril, editors, *The Mathematics of Paul Erdös*, pp. 228–240. Springer-Verlag, 1997.

[23] S. Piccard. *Sur les Bases du Groupe Symétrique et les Couples de Substitutions Qui Engendrent un Groupe Régulier*, Vol. 19 of *Mémoires de l'Université de Neuchâtel*. Librairie Vuibert, 1946.

[24] G. Pighizzini and J. Shallit. Unary language operations, state complexity and Jacobsthal's function. *Internat. J. Found. Comp. Sci.* **13** (2002), 145–159.

[25] J.-E. Pin. *Varieties of Formal Languages*. North Oxford Academic, 1986.

[26] J.-E. Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, Vol. 1, pp. 679–746. Springer-Verlag, 1997.

[27] J.-E. Pin and J. Sakarovitch. Some operations and transductions that preserve rationality. In A. B. Cremers and H. P. Kriegel, editors, *Theoretical Computer Science: Sixth GI-Conference (Dortmund 1983)*, Vol. 145 of *Lecture Notes in Computer Science*, pp. 277–288. Springer-Verlag, 1983.

[28] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.* **3** (1959), 114–125.

[29] R. C. Read. An introduction to chromatic polynomials. *J. Combin. Theory* **4** (1968), 52–71.

[30] H. Robbins. A remark on Stirling's formula. *Amer. Math. Monthly* **62** (1955), 26–29.

[31] A. Salomaa. On basic groups for the set of functions over a finite domain. *Ann. Acad. Sci. Fenn.*, Ser. A I Math. 338, 1963.

[32] A. Salomaa. Composition sequences for functions over a finite domain. *Theoret. Comput. Sci.* **292** (2003), 263–281.

[33] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Develop.* **3** (1959), 198–200.

[34] M. Szalay. On the maximal order in S_n and S_n^*. *Acta Arith.* **37** (1980), 321–331.

[35] S. Yu. State complexity of regular languages. *J. Automata, Languages, and Combinatorics* **6** (2001), 221–324.

[36] S. Yu and Q. Zhuang. On the state complexity of intersection of regular languages. *SIGACT News* **22** (1991), 52–54.

[37] S. Yu, Q. Zhuang, and K. Salomaa. The state complexities of some basic operations on regular languages. *Theoret. Comput. Sci.* **125** (1994), 315–328.

[38] G.-Q. Zhang. Automata, Boolean matrices, and ultimate periodicity. *Inform. Comput.* **152** (1999), 138–154.