Problems Related to Shortest Strings in Formal Languages

by

Thomas Ang

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, 2010

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 formal language theory, studying shortest strings in languages, and variations thereof, can be useful since these strings can serve as small witnesses for properties of the languages, and can also provide bounds for other problems involving languages. For example, the length of the shortest string accepted by a regular language provides a lower bound on the state complexity of the language.

In Chapter 1, we introduce some relevant concepts and notation used in automata and language theory, and we show some basic results concerning the connection between the length of the shortest string and the nondeterministic state complexity of a regular language. Chapter 2 examines the effect of the intersection operation on the length of the shortest string in regular languages. A tight worst-case bound is given for the length of the shortest string in the intersection of two regular languages, and loose bounds are given for two variations on the problem. Chapter 3 discusses languages that are defined over a free group instead of a free monoid. We study the length of the shortest string in a regular language that becomes the empty string in the free group, and a variety of bounds are given for different cases. Chapter 4 mentions open problems and some interesting observations that were made while studying two of the problems: finding good bounds on the length of the shortest squarefree string accepted by a deterministic finite automaton, and finding an efficient way to check if a finite set of finite words generates the free monoid.

Some of the results in this thesis have appeared in work that the author has participated in [3, 4].

Acknowledgements

First, I would like to thank Jeffrey Shallit for his excellent supervision, which has made my experience as a graduate student an extremely positive one. I would also like to thank my other collaborators past and present: John Brzozowski, Giovanni Pighizzini and Narad Rampersad. I am thankful to Jonathan Buss and John Watrous for agreeing to be on my thesis committee. I would also like to acknowledge the fine people of the University of Waterloo for making my extended stay a pleasant one. Finally, I thank the many teachers I have had over the years, both inside and outside the classroom.

Dedication

This is dedicated to my mother who taught me to add two positive integers a long time ago, and who will soon be a university graduate herself.

Contents

List of Figures

Chapter 1

Introduction

1.1 Automata, Languages and Shortest Strings

In this thesis, we discuss several problems arising from automata theory and the theory of formal languages. While we approach the study of automata and formal languages in a mathematical manner, with the field's place as a cornerstone in the theory of computation in mind, the theory has applications in a wide variety of areas including neural networks, switching circuits and lexical analyzers. For a more complete overview of automata and language theory and its applications, see an appropriate text like the one by Hopcroft and Ullman [25]. The problems we consider have a common theme in that they deal with quantitative properties of special automata and languages, most commonly the length of the shortest accepted string. These properties have implications on the efficient use of "resources", which in this case refers not to computational complexity, but to descriptional complexity [23]: essentially the number of symbols needed to describe the objects in question.

In the theory of computation, problems are classified according to whether or not they can be solved by a particular model of computation. We often restrict ourselves to decision problems, which are problems where a given input must be mapped to a value of "yes" or "no". In this case, the solution to any problem can be represented by a set: the set of all inputs for which the the answer to the problem is "yes". In computer science we typically encode data over an *alphabet* of symbols or *letters*, for example 0s and 1s. So

each input that we consider can be thought of as a sequence or *word* made up of symbols from an alphabet. Sometimes we also use the term *string* to refer to a word, and we use the two terms interchangeably. A set of words we call a *language*. Typically we denote the alphabet by Σ, and we let Σ^* be the free monoid generated by Σ. Then a word is an element of Σ^* and a language is a subset of Σ^*. We denote the empty word by ϵ and the empty set by $\emptyset$.

Just as with instances of any other kind of set, we can take the *intersection*, the *union* (in addition to the usual notation we sometimes denote this with $+$) and the *complement* of languages. Since the elements of languages are sequences, we can also generalize some operations that rely on the order of the symbols in sequences to apply to languages. The *reversal* of a word w is the word w^R formed by taking the symbols of w in their reversed order. The reversal of a language L is the language L^R consisting of the reversal of each word in L. The *concatenation* of two words u and v is formed by prepending the first word to the second word, and is denoted uv. The concatenation of two languages K and L is the language of words obtained by prepending a word from the first language to a word from the second, and is denoted by KL. If L is a language and k a positive integer, by L^k, or the k-th *power* of L, we mean the language consisting of words formed by concatenating k words from L. The *Kleene closure* of L, denoted by L^*, is the language

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

and the *positive closure* of L, denoted by L^+, is the language

$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

The power, the Kleene closure and the positive closure can each also be applied to words by treating a word as a singleton language.

Some very simple models of computation are the classes of *finite automata* (FAs). These are systems or machines that make transitions between a finite number of states on the basis of symbols read from input. Furthermore, a *deterministic finite automaton* (DFA) has exactly one transition defined for each pair of state and input symbol. This means that it must behave entirely deterministically and hence its name. A *nondeterministic finite automaton* (NFA), on the other hand, can have any number of possible transitions

defined for each state/input pair so that the machine can essentially choose which state to transition to from some subset of all the states in the machine. A slight extension of the NFA is the ϵ-NFA, in which one can define ϵ-transitions, which are transitions that can be taken without consuming input symbols. An extension that brings us beyond the realm of finite automata is the *pushdown automaton* (PDA), which is basically a finite automaton that also has control of an unbounded amount of extra memory in the form of a stack. There are many other kinds of automata, some of which are explained in the text by Hopcroft and Ullman [25].

The notation we use to describe automata is as follows. We denote a DFA by a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite 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 accepting or final states. We denote an NFA by a similar quintuple except the transition function is $\delta : Q \times \Sigma \to 2^Q$. If ϵ-transitions are allowed, then the transition function is $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$. For all three kinds of finite automata we make a usual convenient generalization on δ to the extended transition function with domain $Q \times \Sigma^*$. If M is deterministic, a word $w \in \Sigma^*$ is *accepted* by M if $\delta(q_0, w) \in F$, otherwise w is accepted by M if $\delta(q_0, w) \cap F \neq \emptyset$.

We denote a PDA by a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q is a finite set of states, Σ is the finite input alphabet, Γ is the finite alphabet of symbols allowed on the stack, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is the initial stack symbol, $F \subseteq Q$ is the set of accepting states, and δ is the transition function that maps from $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$. We represent the configurations of a PDA by triples in $Q \times \Sigma^* \times \Gamma^*$, sometimes called *instantaneous descriptions*. For $a \in \{\epsilon\} \cup \Sigma$, for $p, q \in Q$, for $w \in \Sigma^*$, and for $\alpha, \beta, \gamma \in \Gamma^*$, we say that $(q, aw, \gamma\alpha) \vdash_M (p, w, \beta\alpha)$ if $\delta(q, a, \gamma)$ contains (p, β). We use $\vdash_M^*$ for the reflexive and transitive closure of $\vdash_M$. There are two ways to define acceptance by a PDA, which we will call acceptance by empty stack and acceptance by final state. A word $w \in \Sigma^*$ is accepted by empty stack by M if $(q_0, w, Z_0) \vdash_M^* (p, \epsilon, \epsilon)$ for some $p \in Q$. A word $w \in \Sigma^*$ is accepted by final state by M if $(q_0, w, Z_0) \vdash_M^* (p, \epsilon, \gamma)$ for some $p \in F, \gamma \in \Gamma^*$.

For any automaton M, we let the language $L(M)$ be the set of all words accepted by M. The classes of languages accepted by each of these types of automata have been well studied. The DFAs, NFAs and ϵ-NFAs are all equally powerful and accept what is known as the class of *regular languages* (the set of languages generated by regular expressions),

and the class of languages accepted by PDAs by empty stack is the same as the class of languages accepted by PDAs by final state and is known as the class of *context-free languages* (the set of languages generated by context-free grammars). For more details on regular expressions and context-free grammars, again one may consult the text by Hopcroft and Ullman [25].

Closure properties for both classes have been well studied. By definition, the class of regular languages is closed under union, concatenation and Kleene closure, but it has also been found to be closed under a variety of other operations including complementation, intersection, reversal and homomorphisms. The class of context-free languages, which includes all regular languages, is also closed under union, concatenation, Kleene closure and homomorphisms, but it is not closed under intersection or complementation. Closure properties for a wide variety of other classes of languages have been studied as well, for example the classes of convex languages defined by binary relations such as the *prefix*, *suffix*, *factor* and *subword* relations[1] [2].

In addition to closure properties, there are also quantitative properties of languages that are of interest and they require the following notation in order to be discussed. For any $x \in \Sigma^*$, $|x|$ denotes the length of x, and $|x|_a$ for some $a \in \Sigma$ denotes the number of occurrences of a in x. We let $|\Sigma|$ denote the alphabet size. We define maps from nonempty languages to nonnegative integers as follows. For a nonempty language L, let $\mathrm{lss}(L)$ denote the length of the shortest string in L. If L is regular, then we let $\mathrm{sc}(L)$ denote the state complexity of L (the minimum number of states in any DFA accepting L), and let $\mathrm{nsc}(L)$ denote the nondeterministic state complexity of L (the minimum number of states in any NFA accepting L).

State complexity and nondeterministic state complexity are interesting properties of a language since they give bounds on the amount of space required to solve the problems that correspond to the language. For this reason, this has been a well studied area with numerous results in topics as varied as the impact of basic operations [38], the effect of

[1]These relations appear in this thesis and are defined in the following way. If there exist $x, y, z \in \Sigma^*$ and $w = xyz$, we say that y is a factor of w. If $x = \epsilon$, we also say that y is a prefix of w. If $z = \epsilon$, we also say that y is a suffix of w. We say that y is a subword of w if we can write $y = a_1 a_2 \cdots a_n$ and $w = w_1 a_1 w_2 a_2 \cdots w_n a_n w_{n+1}$ for some letters $a_i \in \Sigma$ and words $w_i \in \Sigma^*$. If y is a prefix, suffix, factor or subword of w and $y \neq w$, then y is a *proper* prefix, proper suffix, proper factor or proper subword of w, respectively.

combined operations on some special languages [24], bounds for finite languages [16], as well as other works, some of which are cited later in this thesis [17, 21, 22, 27, 28, 33, 36]. Recently, Brzozowski has developed an entirely new approach to state complexity that defines this property of regular languages in terms of characteristics of the languages themselves as opposed to the automata that accept them [15]. There are many other results that are not touched upon here.

In previous work, questions have been asked about shortest strings in languages to achieve specific properties. These shortest strings are interesting because they can serve as small witnesses for properties of the languages and can provide bounds for other problems. For example, the length of shortest string not accepted by an NFA was studied by Ellul et al. [17]. The fact that the shortest string in this case is exponential in length, in terms of the number of states, implies that it is not feasible to check by brute force if an NFA accepts Σ^*.

Shortest strings have also been studied as a measure of complexity of languages, specifically in the context of rational indices of languages [11]. This notion is revisited in Section 3.3, and we provide a definition at that point in the thesis. Here we begin with a simpler relationship between shortest strings and complexity: for any nonempty regular language L, the quantities $\mathrm{sc}(L)$ and $\mathrm{nsc}(L)$ are bounded below by $\mathrm{lss}(L)$. This is due to the pumping lemma for regular languages, which is stated in the text by Hopcroft and Ullman [25, p. 56]. We state it with our own conventions as follows.

Theorem 1 *Let L be a regular language. Then there is a positive integer n such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n, |v| \geq 1$, and for all $i \geq 0, uv^iw$ is in L. Furthermore, n is no greater than the number of states of the smallest FA accepting L.*

This theorem holds for both DFAs and NFAs. The precise and tight bounds given by $\mathrm{lss}(L)$ are in the following proposition.

Proposition 2 *For any nonempty regular language L we have $\mathrm{lss}(L) < \mathrm{nsc}(L) \leq \mathrm{sc}(L)$, and for each integer $k \geq 1$ there exists a language L_k such that $\mathrm{lss}(L_k) = \mathrm{nsc}(L_k) - 1$ and $\mathrm{nsc}(L_k) = \mathrm{sc}(L_k)$.*

Proof: The inequality $\text{nsc}(L) \leq \text{sc}(L)$ comes from the fact that every DFA is also an NFA. The inequality $\text{lss}(L) < \text{nsc}(L)$ holds because the pumping lemma tells us that the existence of any string in L of length greater than or equal to the number of states in the minimal FA that accepts L implies that another string of shorter length is also accepted. This is essentially because the shortest string accepted by an FA can visit each state at most once, otherwise the states visited between the first and second visit to a repeated state are unnecessary.

Finally, to see that the bounds are tight, observe that the language $L_k = 0^{k-1}0^*$ over $\Sigma = \{0\}$, for any positive integer k, has $\text{lss}(L_k) = k-1$, has $\text{nsc}(L_k) = k$ and has $\text{sc}(L_k) = k$.

□

The bounds given by the previous proposition are particularly interesting since if L is represented by an NFA or DFA, then there is no known efficient way to compute $\text{nsc}(L)$ [21, 22]. Various techniques have been used to approximate lower bounds on nondeterministic state complexity including two similar techniques described in the following theorem.

Theorem 3 *Let $L \subseteq \Sigma^*$ be a regular language, and suppose there exists a set of pairs $P = \{(x_i, y_i) \in \Sigma^* \times \Sigma^* : 1 \leq i \leq n\}$ such that $x_i y_i \in L$ for $1 \leq i \leq n$. Define the following properties:*

1. *$x_j y_i \notin L$ for $1 \leq i, j \leq n$, and $i \neq j$;*
2. *$x_i y_j \notin L$ or $x_j y_i \notin L$ for $1 \leq i, j \leq n$, and $i \neq j$.*

If P has either property, then any NFA accepting L has at least n states.

The first property is due to Glaister and Shallit [21], and the second is from Birget [10]. While the technique of Glaister and Shallit can be good in practice, sometimes simply looking at the length of the shortest string in a given language can provide an arbitrarily better lower bound on the nondeterministic state complexity.

Proposition 4 *For each integer $k \geq 1$, there exists a regular language L_k for which the lower bound on $\text{nsc}(L)$ given by the technique of Glaister and Shallit is 1 whereas the true bound is $\text{lss}(L) + 1 = k$.*

Proof: Over $\Sigma = \{0\}$, if L_k is the language $0^{k-1}0^*$, one can observe that $\text{nsc}(L_k) = k = \text{lss}(L) + 1$. Now we show that the technique of Glaister and Shallit gives the bound $\text{nsc}(L_k) > 1$ regardless of the value of k. For the sake of contradiction, suppose that for some L_k there exist two pairs (x_1, y_1) and (x_2, y_2) such that $x_1y_1, x_2y_2 \in L_k$ and $x_1y_2, x_2y_1 \notin L_k$. Then $x_1 = 0^i, y_1 = 0^j, x_2 = 0^m, y_2 = 0^n$ for some integers $i, j, m, n \geq 0$ such that $i+j \geq k-1$ and $m + n \geq k - 1$, while $i + n < k - 1$ and $j + m < k - 1$. However, combining these inequalities gives us that $i + j + m + n \geq 2k - 2$ and $i + j + m + n < 2k - 2$, so this is impossible and by contradiction the proposition is true. □

On the other hand, there are also cases where the technique of Glaister and Shallit finds the true bound while the length of the shortest string does arbitrarily badly in comparison.

Proposition 5 *For each integer $k \geq 1$, there exists a regular language L_k for which the lower bound on* $\text{nsc}(L)$ *provided by the technique of Glaister and Shallit is the true bound of k, while the bound given by* $\text{lss}(L) + 1$ *is 1.*

Proof: Over $\Sigma = \{0\}$, if L_k is the language $(0^k)^*$, then one can observe that $\text{nsc}(L_k) = k$ while $\text{lss}(L_k) + 1$ gives the bound $\text{nsc}(L_k) > 1$ regardless of the value of k. Now we show that the technique of Glaister and Shallit gives the true bound for all values of k. For any given k, for each $1 \leq i \leq k$ let $P_i = (x_i, y_i)$ where $x_i = 0^i$ and $y_i = 0^{k-i}$. Then it is easy to verify that we have a set of k pairs that satisfy the conditions required by Theorem 3, and that hence by the theorem we have the tight bound $\text{nsc}(L_k) \geq k$. □

One might observe that the length of the shortest nonempty string provides a bound that is always at least as good as the one given by the length of the shortest string; however, by modifying the proof of the previous proposition, we can see that in the worst case the improvement is insignificant: over $\Sigma = \{0\}$, let L_k be $0(0^k)^*$, then the length of the shortest nonempty string gives us a bound of 2 regardless of k while the technique of Glaister and Shallit still gives the true bound. The technique of Birget is stronger than that of Glaister and Shallit, and there are no languages for which the bound on nondeterministic state complexity given by examining the length of the shortest string is better than the bound given by the technique of Birget.

Proposition 6 *If L is a regular language with* $\text{lss}(L) = n - 1$, *then there exists a set of n pairs $P = \{(x_i, y_i) \in \Sigma^* \times \Sigma^* : 1 \leq i \leq n\}$ such that $x_iy_i \in L$ for $1 \leq i \leq n$, and $x_iy_j \notin L$ or $x_jy_i \notin L$ for $1 \leq i, j \leq n$, and $i \neq j$.*

Proof: Let w be the shortest string in L, which is of length $n-1$. Define $w_{i,j}$ to be the factor of w that begins at the ith position in w and ends at the jth position in w. If $i = j$ then $w_{i,j}$ is the ith letter in w, and if $i > j$ then $w_{i,j} = \epsilon$. For each $0 \leq i < n$ let $P_i = (x_i, y_i)$, where $x_i = w_{1,i}$ and $y_i = w_{i+1,|w|}$. Then it is easy to verify that for any integers $1 \leq i, j \leq n$, we have $x_i y_i \in L$ and we have either $|x_i y_j| < \mathrm{lss}(L)$ or $|x_j y_i| < \mathrm{lss}(L)$. So we have n pairs that meet the requirements of Birget's technique to show that $\mathrm{nsc}(L) \geq n$. □

This thesis has two main parts as outlined by the remainder of the introduction, and then finishes by examining some open problems in Chapter 4 that were not solved during the course of the author's studies. Chapter 2 is centred around a new tight bound for the length of the shortest string in the intersection of two regular languages, and takes a look at two related problems. Chapter 3 is focused on a variety of shortest string bounds that arise when we define languages over a free group instead of a free monoid.

1.2 The Intersection of Regular Languages

Rabin and Scott [33] observed that the state complexity of the intersection of two regular languages that have state complexities m and n has an upper bound of mn. One can easily verify this result using the usual cross-product construction [25, p. 59]. From Proposition 2 we have that the shortest word in such an intersection cannot be longer than $mn-1$. It is natural to wonder if this bound is the best possible over a fixed alphabet size for every choice of m and n.

Over a one-letter alphabet, if we restrict ourselves to values of m and n such that $\gcd(m, n) = 1$, then there is an obvious construction that gives a tight bound.

Proposition 7 *For all integers $m, n \geq 1$ such that $\gcd(m, n) = 1$, there exist DFAs M_1, M_2 with m and n states, respectively, and with $|\Sigma| = 1$ such that $L(M_1) \cap L(M_2) \neq \emptyset$, and $\mathrm{lss}(L(M_1) \cap L(M_2)) = mn - 1$.*

Proof: Let M_1 be the m-state automaton that accepts $L(M_1) = \{x : |x| \equiv m-1 \pmod{m}\}$, and let M_2 be the n-state automaton that accepts $L(M_2) = \{x : |x| \equiv n-1 \pmod{n}\}$. Since $\gcd(m, n) = 1$, we have $L(M_1) \cap L(M_2) = \{x : |x| \equiv mn-1 \pmod{mn}\}$, and it follows that $\mathrm{lss}(L(M_1) \cap L(M_2)) = mn - 1$. □

However, when this is generalized to arbitrary values for m and n, the bound is not tight as the difference between $mn-1$ and the true bound varies proportionally with Jacobsthal's function [36]. In order to achieve a tight bound for the general case, a two-letter alphabet is sufficient. This is a new result shown in Section 2.2. Section 2.3 examines the case where we take the positive closure of the two languages before taking their intersection. Finally, Section 2.4 discusses generalizing the problem to the intersection of an arbitrary number of languages.

1.3 Automata and Languages over Free Groups

A word in a free group can be represented in many different ways. For example, aaa^{-1} and $aa^{-1}baa^{-1}b^{-1}a$ are two different ways to write the word a. Among all the different representations, however, there is one containing no occurrences of a letter next to its own inverse. Following Berstel [7], we call such a representation *reduced*. Though we do first review some closure properties and state complexity results for some operations related to *reduced words*, in this part of the thesis we concern ourselves primarily with new results for length of shortest string problems. The type of problem that we study is actually is special case of the problem of rational indices of context-free languages, studied by Boasson et al. [11]. Other special cases of this problem have been examined by Pierre and Farinone [29]. Some other questions about formal languages and reduced representations have previously been studied in contexts such as rational subsets of free groups [5], Dyck languages [7], automatic groups [18], and string rewriting systems [6, 12, 13, 14]. The references cited here are by no means a complete list of works in the area, but rather are a small sample of the breadth of related studies.

In addition to the standard notation introduced in Section 1.1, we also define some notation specific to our problem. For a letter a, we denote its inverse by a^{-1}, and we let the empty word, ϵ, be the identity. The inverse of a letter is unique and $(a^{-1})^{-1} = a$. We consider only alphabets of the form $\Sigma = \Gamma \cup \Gamma^{-1}$, where, for a positive integer k, $\Gamma = \{1, 2, \ldots, k\}$ and $\Gamma^{-1} = \{1^{-1}, 2^{-1}, \ldots, k^{-1}\}$. For a word $w = a_1 a_2 \cdots a_n \in \Sigma^*$, we denote its inverse by $w^{-1} = a_n^{-1} \cdots a_2^{-1} a_1^{-1}$, and for a language $L \subseteq \Sigma^*$, we let $L^{-1} = \{w^{-1} : w \in L\}$. Note that taking the inverse of a word is equivalent to reversing it and then applying a homomorphism that maps each letter to its inverse. Now, we introduce a

reduction operation on words, consisting of removing factors of the form aa^{-1}, with $a \in \Sigma$. More formally, let us define the relation $\vdash \subseteq \Sigma^* \times \Sigma^*$ such that, for all $w, w' \in \Sigma^*$, $w \vdash w'$ if and only if there exists $x, y \in \Sigma^*$ and $a \in \Sigma$ satisfying $w = xaa^{-1}y$ and $w' = xy$. As usual, $\vdash^*$ denotes the reflexive and transitive closure of $\vdash$.

As is further discussed in Section 3.1 (see Theorem 15), it turns out that for each $w \in \Sigma^*$ there exists exactly one word $r(w) \in \Sigma^*$ such that $r(w)$ does not contain any factor of the form aa^{-1}, with $a \in \Sigma$, and $w \vdash^* r(w)$. We call this $r(w)$ the *reduced representation* of w, and it can be obtained from w by repeatedly replacing with ϵ all factors of the form aa^{-1}, for any letter $a \in \Sigma$, until no such factor exists. If $w = r(w)$ we say that w is a reduced word. If $r(w) = \epsilon$ we say that w is *ϵ-reducible*. We can extend the reduced representation to languages so that for $L \subseteq \Sigma^*$, we let $r(L) = \{r(w) : w \in L\}$. The reduced representation of any language is also unique, and the reduced representation of the free monoid generated by $\Sigma = \Gamma \cup \Gamma^{-1}$ is the free group generated by Γ.

Given a language L, the $\vdash$-closure of L is the set of the words which can be obtained by applying the operation $\vdash$ repeatedly to all words of L, i.e., the set $\{x \in \Sigma^* : \exists w \in L \text{ s.t. } w \vdash^* x\}$. For $w \in \Sigma^*$, the *set of equivalent words* is the language $\mathrm{eq}(w) = \{w' : r(w) = r(w')\}$. For $L \subseteq \Sigma^*$, the set of equivalent words is $\mathrm{eq}(L) = \{\mathrm{eq}(w) : w \in L\}$. Notice that $r(L)$ coincides with the intersection of the $\vdash$-closure of L and $r(\Sigma^*)$, and also with $\mathrm{eq}(L) \cap r(\Sigma^*)$. A set of equivalent words can be thought of as an equivalence class under an equivalence relation described by a very particular set of equations: for all $a \in \Sigma, aa^{-1} = \epsilon$. When this *defining set of equations* is generalized to allow for an arbitrary set of equations we are dealing with what those who study string rewriting systems refer to as a *Thue system*.

Section 3.1 examines the reduced representations and equivalent sets of words for regular and context-free languages. Section 3.2 gives some bounds on the state complexity of reduced representations. In Section 3.3 we look at bounds on the length of the shortest ϵ-reducible word in a regular language in terms of the nondeterministic state complexity of the language. Section 3.4 gives bounds on the length of shortest ϵ-reducible words over restricted alphabets.

Chapter 2

The Intersection of Regular Languages

2.1 The Cross-Product Construction

The bound of mn on the state complexity of the intersection of two regular languages with state complexities m and n can be verified with the cross-product construction, which is sketched with DFAs in the text by Hopcroft and Ullman [25, p. 59], but which also works with NFAs as follows.

Theorem 8 *Given two NFAs M_1 and M_2 with m and n states, respectively, there exists an NFA M with mn states that accepts $L(M) = L(M_1) \cap L(M_2)$.*

Proof: The idea is that to accept the intersection of the languages accepted by two NFAs, M_1 and M_2, we create a new NFA M that simulates the two NFAs in parallel. If $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, then

$$M = (Q_1 \times Q_2, \Sigma, \delta, [q_1, q_2], F_1 \times F_2),$$

where for all $p_1 \in Q_1, p_2 \in Q_2, a \in \Sigma$,

$$\delta([p_1, p_2], a) = \delta_1(p_1, a) \times \delta_2(p_2, a).$$

□

To generalize the construction shown in the proof of Proposition 7, we construct automata based on integers $j \leq m, k \leq n$ such that $\gcd(j,k) = 1$ and the value $jk - 1$ is maximized. Shallit [36] determined that this is the best one can do over a one-letter alphabet by showing that a tight bound on the state complexity of the intersection of two regular languages over a unary alphabet is demonstrated by such a construction. The general idea is based on the observation that any unary DFA is composed of a 'tail' and a 'cycle'. If we intersect two DFAs, the lowest common multiple of the cycle size in each is the dominant term in the number of distinct states in the cross-product construction.

2.2 A Tight Bound over a Two-Letter Alphabet

Here is a construction found by the author over a two-letter alphabet that achieves a tight bound of $mn-1$ for the length of the shortest string found in the intersection of two regular languages.

Theorem 9 *For all integers $m, n \geq 1$ there exist DFAs M_1, M_2 with m and n states, respectively, and with $|\Sigma| = 2$ such that $L(M_1) \cap L(M_2) \neq \emptyset$, and $\mathrm{lss}(L(M_1) \cap L(M_2)) = mn - 1$.*

Proof: The proof is constructive. Without loss of generality, assume $m \leq n$, and set $\Sigma = \{0,1\}$. Let M_1 be the DFA given by $(Q_1, \Sigma, \delta_1, p_0, F_1)$, where $Q_1 = \{p_0, p_1, p_2, \ldots, p_{m-1}\}$, $F_1 = p_0$, and for each a, $0 \leq a \leq m-1$, and $c \in \{0,1\}$ we set

$$\delta_1(p_a, c) = p_{(a+c) \bmod m}.$$

Then

$$L(M_1) = \{x \in \Sigma^* : |x|_1 \equiv 0 \pmod{m}\}.$$

Let M_2 be the DFA $(Q_2, \Sigma, \delta_2, q_0, F_2)$, shown in Figure 2.1, where $Q_2 = \{q_0, q_1, \ldots, q_{n-1}\}$, $F_2 = q_{n-1}$, and for each a, $0 \leq a \leq n-1$,

$$\delta_2(q_a, c) = \begin{cases} q_{a+c}, & \text{if } 0 \leq a < m-1; \\ q_{(a+1) \bmod n}, & \text{if } c = 0 \text{ and } m-1 \leq a \leq n-1; \\ q_0, & \text{if } c = 1 \text{ and } m-1 \leq a \leq n-1. \end{cases}$$

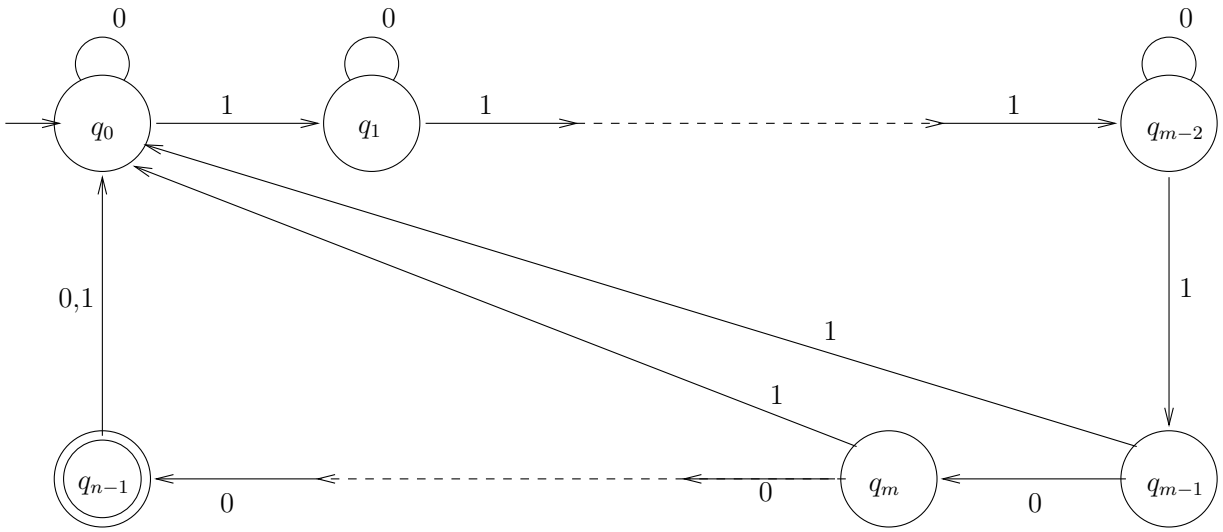

Figure 2.1: The DFA M_2.

Focussing solely on the 1's that appear in some accepting path in M_2, we see that we can return to q_0

(a) via a simple path with m 1's, or

(b) (if we go through q_{n-1}), via a simple path with $m-1$ 1's and ending in the transition $\delta(q_{n-1}, 0) = q_0$.

After some number of cycles through q_0, we eventually arrive at q_{n-1}. Letting i denote the number of times a path of type (b) is chosen (including the last path that arrives at q_{n-1}) and j denote the number of times a path of type (a) is chosen, we see that the number of 1's in any accepted word must be of the form $i(m-1)+jm$, with $i > 0$, $j \geq 0$. The number of 0's along such a path is then at least $i(n-m+1)-1$, with the -1 in this expression arising from the fact that the last part of the path terminates at q_{n-1} without taking an additional 0 transition back to q_0.

Thus

$$L(M_2) \subseteq \{x \in \Sigma^* : \exists i, j \in \mathbb{N}, \text{ such that } i > 0, j \geq 0, \text{ and } |x|_1 = i(m-1)+jm,\ |x|_0 \geq i(n-m+1)-1\}.$$

Furthermore, for every $i, j \in \mathbb{N}$, such that $i > 0, j \geq 0$, there exists an $x \in L(M_2)$ such that $|x|_1 = i(m-1) + jm$, and $|x|_0 = i(n-m+1) - 1$. This is obtained, for example, by cycling j times from q_0 to q_{m-1} and then back to q_0 via a transition on 1, then $i-1$ times from q_0 to q_{n-1} and then back to q_0 via a transition on 0, and finally one more time from q_0 to q_{n-1}.

It follows then that

$$\begin{aligned} L(M_1 \cap M_2) \subseteq \{x \in \Sigma^* : \exists i, j \in \mathbb{N}, \text{ such that } i > 0, j \geq 0, \text{ and} \\ |x|_1 = i(m-1) + jm, \ |x|_0 \geq i(n-m+1) - 1 \\ \text{and } i(m-1) + jm \equiv 0 \pmod{m}\}. \end{aligned}$$

Further, for every such i and j, there exists a corresponding element in $L(M_1 \cap M_2)$. Since $m-1$ and m are relatively prime, the shortest such word corresponds to $i = m$, $j = 0$, and satisfies $|x|_0 = m(n-m+1) - 1$. In particular, a shortest accepted word is $(1^{m-1}0^{n-m+1})^{m-1}1^{m-1}0^{n-m}$, which is of length $mn - 1$. □

2.3 The Intersection of Positive Closures

An interesting related problem arises when we take the positive closures of the two languages in question before taking the intersection. More precisely, given an n-state DFA M_1 and an m-state DFA M_2, what bounds are there on $\text{lss}(L(M_1)^+ \cap L(M_2)^+)$? One might notice that $L(M_1)^+ \cap L(M_2)^+ \supseteq L(M_1) \cap L(M_2)$ and hence expect that $\text{lss}(L(M_1)^+ \cap L(M_2)^+) \leq \text{lss}(L(M_1) \cap L(M_2))$; however, this reasoning does not hold if $L(M_1) \cap L(M_2) = \emptyset \neq L(M_1)^+ \cap L(M_2)^+$.

Shallit [37] observed that the same upper bound of $mn - 1$ holds for $\text{lss}(L(M_1)^+ \cap L(M_2)^+)$:

Proposition 10 *Given two DFAs M_1 and M_2 of m and n states, respectively, if $(L(M_1)^+ \cap L(M_2)^+) \neq \emptyset$, then $\text{lss}(L(M_1)^+ \cap L(M_2)^+) \leq mn - 1$.*

Proof: Observe that given any DFA M, we can construct an ϵ-NFA M' such that $L(M') = L(M)^+$ by adding an ϵ-transition from each final state to the initial state. By following this

procedure, if we have an m-state DFA M_1 and an n-state DFA M_2, we can obtain an m-state ϵ-NFA M_1', and an n-state ϵ-NFA M_2' such that $L(M_1') = L(M_1)^+$ and $L(M_2') = L(M_2)^+$. Then by carrying out the cross-product construction on M_1' and M_2' we get an ϵ-NFA with mn states that accepts $L(M_1)^+ \cap L(M_2)^+$ (see Figure 2.2 for an example). Since it is well known that any ϵ-NFA can be converted to an NFA with the same number of states, our upper bound is given by this construction and Proposition 2. □

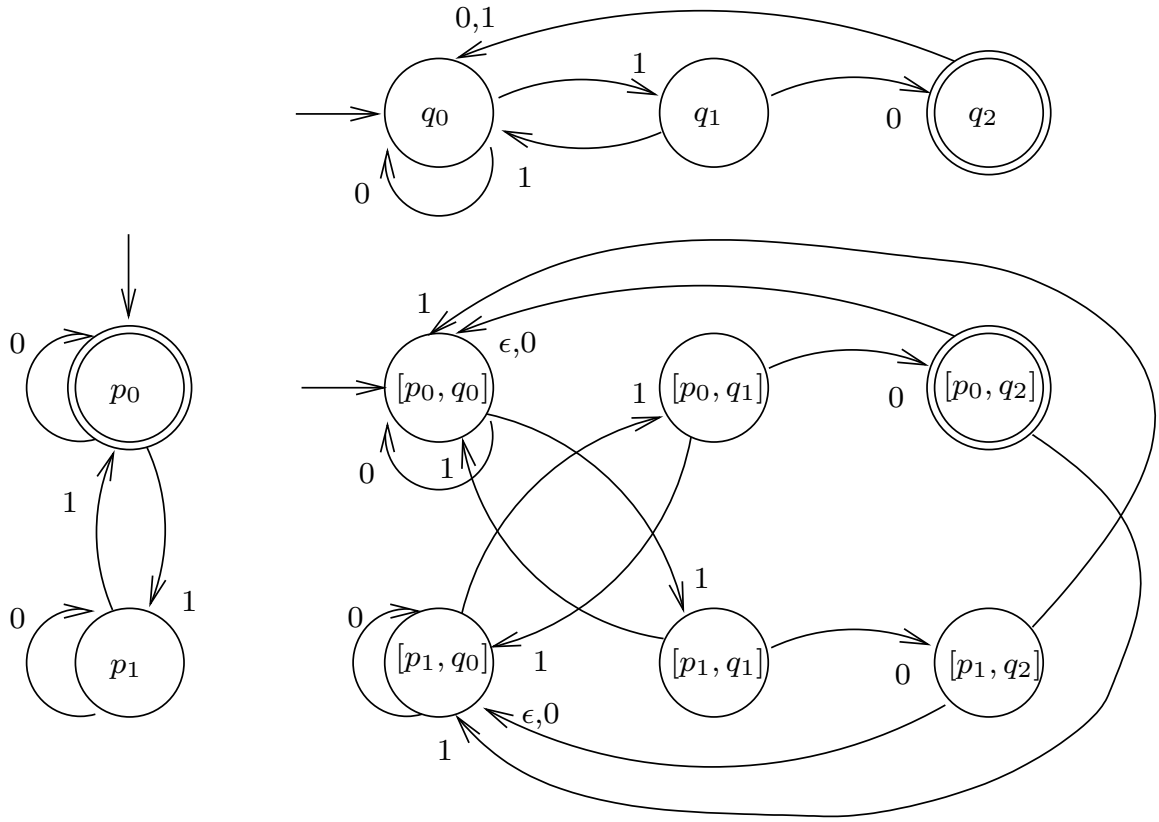

Figure 2.2: Examples for $m = 2$ and $n = 3$. Top: The DFA M_2. Bottom-left: The DFA M_1. Bottom-right: An ϵ-NFA with mn states that accepts $L(M_1)^+ \cap L(M_2)^+$.

One can see in Figure 2.2 that, due to the addition of the ϵ-transitions, the construction from Theorem 9 for $m = 2$ and $n = 3$ accepts a shortest string of 1010 instead of 10010. In general the construction from Theorem 9 only achieves a shortest string of length $mn - m$ when the positive closure is applied before the intersection, and the author has been unable

to find any example that gives a better lower bound. The next theorem shows that the upper bound from Proposition 10 is not tight, and implies that the lower bound of $mn-m$ is tight for $m = 2$.

Theorem 11 *For any m-state DFA M_1 and n-state DFA M_2 such that $L(M_1)^+ \cap L(M_2)^+ \neq \emptyset$ we have $\mathrm{lss}(L(M_1)^+ \cap L(M_2)^+) < mn - 1$.*

Proof: For the sake of contradiction, assume we have DFAs M_1 and M_2 with m and n states, respectively, such that $\mathrm{lss}(L(M_1)^+ \cap L(M_2)^+) = mn - 1$. Let M_1 be the DFA given by $(Q_1, \Sigma, \delta_1, p_0, F_1)$, where $Q_1 = \{p_0, p_1, p_2, \ldots, p_{m-1}\}$, and let M_2 be the DFA given by $(Q_2, \Sigma, \delta_2, p_0, F_2)$, where $Q_2 = \{q_0, q_1, q_2, \ldots, q_{n-1}\}$. Then let M_1' and M_2' be the ϵ-NFAs obtained by adding ϵ-transitions from the final states to the start states in M_1 and M_2, respectively. Let M be the ϵ-NFA obtained by applying the cross-product construction to M_1' and M_2'. Then M accepts $L(M_1)^+ \cap L(M_2)^+$.

If M has more than one final state, a shortest accepting path would only visit one of them, and this immediately gives a contradiction. So, assume each of M_1 and M_2 have only one final state; that is $F_1 = \{p_x \in Q_1\}$ and $F_2 = \{q_y \in Q_2\}$. Then $M = (Q_1 \times Q_2, \Sigma, \delta, [p_0, q_0], [p_x, q_y])$, where for all $p_i \in Q_1, q_j \in Q_2, a \in \Sigma, \delta([p_i, q_j], a) = [\delta_1(p_i, a), \delta_2(q_j, a)]$. Note that M has ϵ-transitions from $[p_x, q_j]$ to $[p_0, q_j]$ for all $q_j \in Q_2$ and $[p_i, q_y]$ to $[p_i, q_0]$ for all $p_i \in Q_1$.

Let w_1 be a shortest word accepted by M_1 and w_2 be a shortest word accepted by M_2. Then $\delta([p_0, q_0], w_1) = [p_x, q_i]$ for some i such that $q_i \in Q_2$, and while carrying out this computation we never pass through two states $[p_a, q_b]$ and $[p_c, q_d]$ such that $a = c$. Likewise, $\delta([p_0, q_0], w_2) = [p_j, q_y]$ for some j such that $p_j \in Q_1$, and while carrying out this computation we never pass through two states $[p_a, q_b]$ and $[p_c, q_d]$ such that $b = d$. If both $x = 0$ and $y = 0$ the shortest accepted string is ϵ, so without loss of generality, assume $x \neq 0$. Then $\delta([p_0, q_0], w_1) = [p_x, q_0]$ or else we can visit $|w_1| + 2$ states with $|w_1|$ symbols by using an ϵ-transition and we get a contradiction. If $y = 0$, w_1 is the shortest string accepted by M and we have a contradiction. So, $y \neq 0$ and $\delta([p_0, q_0], w_2) = [p_0, q_y]$. It follows that reading w_1 from the initial state brings us to $[p_x, q_0]$ without passing through $[p_0, q_y]$, and reading w_2 from the initial state brings us to $[p_0, q_y]$ without passing through $[p_x, q_0]$. So, a shortest accepting path need only visit one of $[p_x, q_0]$ and $[p_0, q_y]$, and again we have a contradiction. □

While the true bound must be quadratic, since it remains to be found we ask the following question:

Open Problem 12 *What is the greatest integer function $f(m,n)$ such that for all pairs of integers $m,n \geq 1$ we can find DFAs M_1 and M_2 such that $\mathrm{sc}(L(M_1)) = m$, $\mathrm{sc}(L(M_2)) = n$ and $\mathrm{lss}(L(M_1)^+ \cap L(M_2)^+) \geq f(m,n)$?*

The construction from Theorem 9 and the proof from Theorem 11 narrow down the possibilities so that $mn - m \leq f(m,n) \leq mn - 2$.

2.4 The Intersection of More than Two Automata

It is natural to try to generalize the bound from the intersection of two automata to an arbitrary number of DFAs. However, Shallit [4] has found empirically that, over a two-letter alphabet, the corresponding bound $mnp - 1$ for three DFA's does not always hold. For example, there are no DFA's of $2, 2$, and 3 states over a binary alphabet for which the shortest word in the intersection is of length $2 \cdot 2 \cdot 3 - 1$.

With an alphabet size that grows with the number of automata being intersected, it is easy to get a bound that is a polynomial of order equal to the number of automata.

Proposition 13 *For all sets of $n \geq 3$ integers $m_1, m_2, \ldots, m_n \geq 1$ there exist DFAs $M_1, M_2, \ldots, M_n$ with $m_1, m_2, \ldots, m_n$ states, respectively, and with $|\Sigma| = n$ such that $\bigcap_{i=1}^{n} M_i \neq \emptyset$, and $\mathrm{lss}(\bigcap_{i=1}^{n} M_i) = (m_1 m_2 - 1)\Pi_{i=3}^{n}(m_i - 1)$.*

Proof: To show that this proposition holds, we outline a procedure to generate the required DFAs. Let $\Sigma = \{0, 1, \ldots, n-1\}$. For M_1 and M_2, use the DFAs from the construction in the proof of Theorem 9. Then have each M_i for $3 \leq i \leq n$ be the m_i-state DFA that accepts the language $((0 + 1 + \ldots + i - 2)(i-1)^{m_i - 2})^*$. We accomplish this by defining $M_i = (Q_i, \Sigma, \delta_i, q_{i,0}, F_i)$, where $Q_i = \{q_{i,0}, q_{i,1}, q_{i,2}, \ldots, q_{i,m_i-1}\}$, $F_i = q_{i,0}$, and for each a, $0 \leq a \leq m_i - 2$, and $c \in \Sigma$ we set

$$\delta_i(q_{i,a}, c) = \begin{cases} q_{i,1}, & \text{if } a = 0 \text{ and } c < i-1; \\ q_{i,(a+1) \bmod m_i - 1}, & \text{if } c = i-1; \\ q_{i,a}, & \text{if } c > i-1. \end{cases}$$

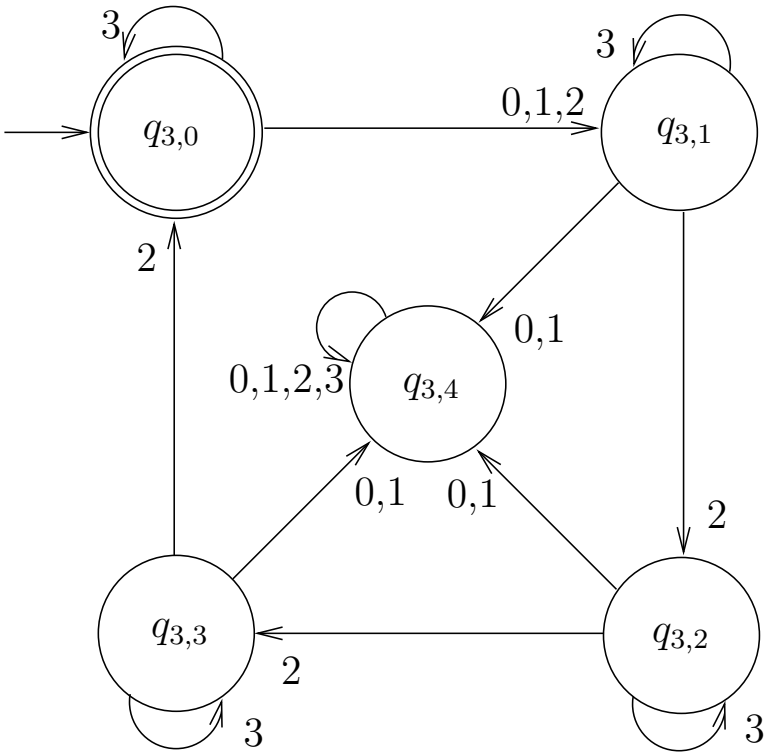

Figure 2.3: The DFA M_3 for $m_3 = 5$ and $n = 4$.

All undefined transitions go to the rejecting state q_{i,m_i-1}. An example is illustrated in Figure 2.3.

For $3 \leq i \leq n$ we define $X_{i,n} = ((i-1)X_{i+1,n})^{m_i-1}$ and for $i > n$ we define $X_{i,n} = \epsilon$. Now we show by induction that for all $n \geq 3$ the shortest string accepted by $\bigcap_{i=1}^{n} M_i$ is

$$w_n = ((1X_{3,n})^{m_1-1}(0X_{3,n})^{m_2-m_1+1})^{m_1-1}(1X_{3,n})^{m_1-1}(0X_{3,n})^{m_2-m_1}.$$

Observe that when $n = 3$, for the shortest accepted string we get

$$((12^{m_3-1})^{m_1-1}(02^{m_3-1})^{m_2-m_1+1})^{m_1-1}(12^{m_3-1})^{m_1-1}(02^{m_3-1})^{m_2-m_1}.$$

Now assume that for some $k \geq 3$ we have the shortest string accepted by $\bigcap_{i=1}^{k} M_i$ as

$$w_k = ((1X_{3,k})^{m_1-1}(0X_{3,k})^{m_2-m_1+1})^{m_1-1}(1X_{3,k})^{m_1-1}(0X_{3,k})^{m_2-m_1}.$$

Then we get the shortest string accepted by $\bigcap_{i=1}^{k+1} M_i$ from taking w_k and inserting $k^{m_{k+1}-1}$ between each pair of adjacent letters. So by induction we have that the shortest string accepted by $\bigcap_{i=1}^{n} M_i$ is

$$w_n = ((1X_{3,n})^{m_1-1}(0X_{3,n})^{m_2-m_1+1})^{m_1-1}(1X_{3,n})^{m_1-1}(0X_{3,n})^{m_2-m_1},$$

which is of length $(m_1m_2 - 1)\Pi_{i=3}^{n}(m_i - 1)$. □

Unfortunately, a tight lower bound on the length of the shortest string in the intersection of more than two DFAs has not yet been found, so we have the following problem:

Open Problem 14 *What is the greatest integer function $f(S)$ such that for any finite set of positive integers $S = \{m_1, m_2, \ldots, m_n\}$ we can find DFAs $M_1, M_2, \ldots, M_n$ such that for $1 \leq i \leq n$, we have $\mathrm{sc}(L(M_i)) = m_i$ and $\mathrm{lss}(\bigcap_{i=1}^{n} L(M_i)) \geq f(S)$?*

Chapter 3

Automata and Languages over Free Groups

3.1 Reduced Representations and Sets of Equivalent Words

Here we review some basic results concerning the closure of reduced representations and sets of equivalent words. We examine both regular and context-free languages under these two operations, and show that the behaviour of these operations in our special case is indeed different from their behaviour in general Thue systems.

We begin by mentioning the following theorem that states that the reduced representation of any word is unique. This is actually the case for an entire class of Thue systems that are referred to as having the *Church-Rosser* property, which is discussed, for example, by Book [13]. This theorem is proven in a stronger form by Book and Otto [14].

Theorem 15 *For each $w \in \Sigma^*$ there exists exactly one word $r(w) \in \Sigma^*$ such that $r(w)$ does not contain any factor of the form aa^{-1}, with $a \in \Sigma$, and $w \stackrel{*}{\vdash} r(w)$.*

If we consider arbitrary Thue systems, as shown by the next proposition, it is no longer necessary that reduced representations be unique. Under this generalization, a reduced representation of a word is an equivalent word such that there are no shorter equivalent

words. If g is our defining set of equations, and w is a word, then we denote the set of equivalent words to w under g by $\mathrm{eq}_g(w)$, and denote the set of generalized reduced representations by $r_g(w)$.

Proposition 16 *There exists a set of defining equations g and a word w such that $r_g(w)$ contains more than one word.*

Proof: This example proves the proposition. Let $\Sigma = \{a, b, c, d\}$, $g = \{ab = cd, bc = a\}$, and let $w = bcbd$. Then $\mathrm{eq}(w) = \{abd, cdd, bcbd\}$ and $r_g(w) = \{abd, cdd\}$. □

We now proceed to show that regular languages are closed under the reduction operator by showing that for any regular language L, we can construct $r(L)$ by taking the intersection of $r(\Sigma^*)$ and the $\vdash$-closure of L, both of which are regular. The following lemma shows that $r(\Sigma^*)$ is regular.

Lemma 17 *For $\Sigma = \Gamma \cup \Gamma^{-1}$, where, for a positive integer k, $\Gamma = \{1, 2, \ldots, k\}$ and $\Gamma^{-1} = \{1^{-1}, 2^{-1}, \ldots, k^{-1}\}$, there exists a DFA M_k of $2k + 2$ states that accepts $r(\Sigma^*)$.*

Proof: Recall that a word w is reduced if and only if it does not contain the factor aa^{-1}, for each $a \in \Sigma$. This condition can be verified by defining an automaton M_k that remembers in its finite control the last input letter. To this aim, the automaton has a state q_a for each $a \in \Sigma$. If in the state q_a the symbol a^{-1} is received, then the automaton reaches a dead state q_{k+1}.

Formally, M_k is the DFA $(Q, \Sigma, \delta, q_0, F)$ defined as follows (see Figure 3.1 for an example): $Q = \{q_0, q_{k+1}\} \cup \{q_i : i \in \Gamma \cup \Gamma^{-1}\}$, $F = Q \setminus \{q_{k+1}\}$, and

$$\delta(q_a, c) = \begin{cases} q_c, & \text{if } a \neq c^{-1} \text{ and } q_a \neq q_{k+1}; \\ q_{k+1}, & \text{otherwise.} \end{cases}$$

□

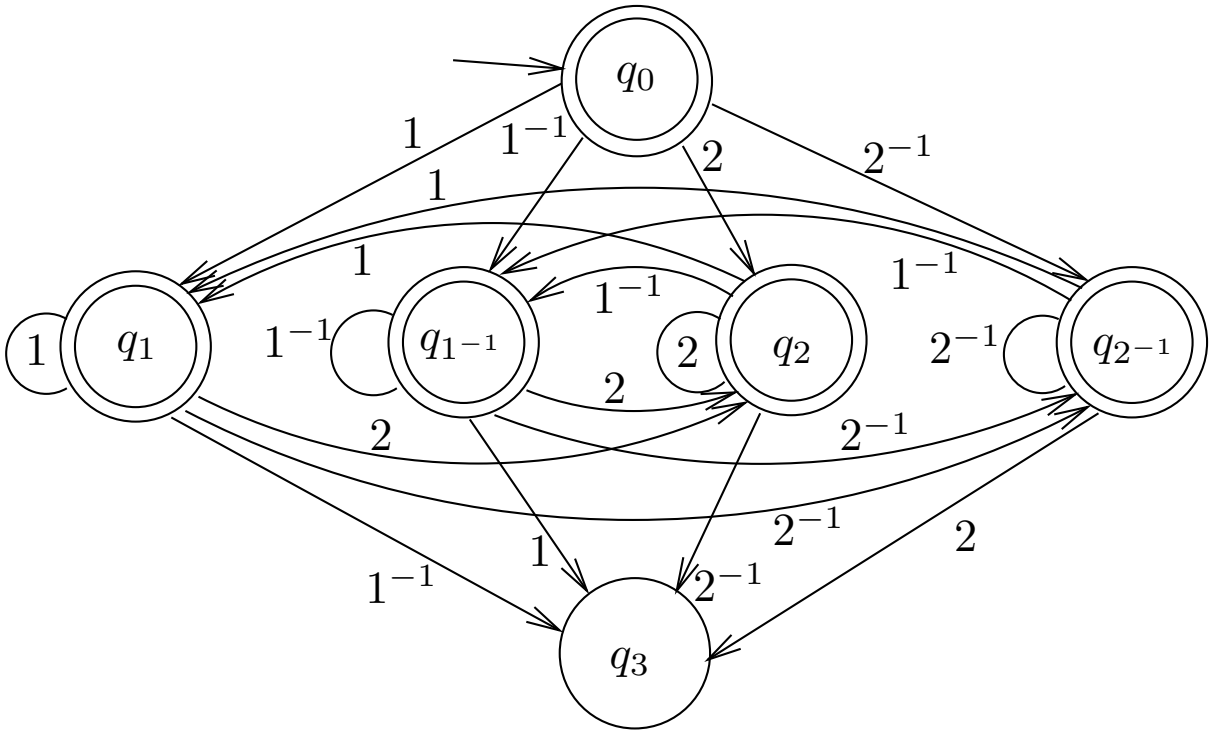

Figure 3.1: M_2, a DFA that accepts $r(\Sigma^*)$ for $k = 2$.

The following theorem shows that the $\vdash$-closure of a regular language is regular. The algorithm in the proof was discovered by Book and Otto [14]. An improved algorithm was found by Benois and Sakarovitch [6].

Theorem 18 *Given an ϵ-NFA $M = (Q, \Sigma, \delta, q_0, F)$ with n states, an automaton M' accepting the $\vdash$-closure of $L(M)$ can be built in $O(n^4)$ time.*

Proof: The idea behind the proof is to present an algorithm that given M computes M' by adding to M ϵ-transitions corresponding to paths on ϵ-reducible words. The algorithm is similar to a well-known algorithm for minimizing DFAs [25, p. 70]. It uses a directed graph $G = (Q, E)$ to remember ϵ-transitions. For each pair of states s, t, the algorithm also keeps a set $l(s, t)$ of pairs of states, with the following meaning: if $(p, q) \in l(s, t)$ and the algorithm discovers that there is a path from s to t on an ϵ-reducible word (and hence it adds the edge (s, t) to G), then there exists a path from p to q on an ϵ-reducible word (thus, the algorithm can also add the edge (p, q)).

$E \leftarrow$ transitive closure of $\{(p,q) \mid q \in \delta(p,\epsilon)\}$
for $s,t \in Q$ **do** $l(s,t) \leftarrow \emptyset$
for $p,q,s,t \in Q$ **do**
  if $\exists a \in \Sigma$ s.t. $s \in \delta(p,a)$ and $q \in \delta(t,a^{-1})$ **then**
    if $(s,t) \in E$ **then** $E \leftarrow update(E,(p,q))$
      else $l(s,t) \leftarrow l(s,t) \cup \{(p,q)\}$

The subroutine *update* returns the smallest set E' having the following properties:

- $E \cup \{(p,q)\} \subseteq E'$;
- if $(p',q') \in E'$ then each element belonging to $l(p',q')$ is in E';
- the graph (Q,E') is transitive.

At the end of the execution, the automaton M' is obtained by adding to M an ϵ-transition from a state p to a state q for each edge (p,q) in the resulting graph G.

Now, we show that the language accepted by M' is the $\vdash$-closure of $L(M)$. To this aim, we observe that for all $p,q,s,t \in Q$, such that $s \in \delta(p,a)$ and $q \in \delta(t,a^{-1})$ are transitions of M for some $a \in \Sigma$, if M' contains an ϵ-transition from s to t then it must contain also an ϵ-transition from p to q. In fact, when the algorithm examines these 4 states in the loop, if (s,t) is in E then the algorithm calls *update* to add (p,q) to E. Otherwise, the algorithm adds (p,q) to $l(s,t)$. Since M' finally contains the ϵ-transition from s to t, then there is a step of the algorithm, after the insertion of (p,q) in $l(s,t)$, adding the pair (s,t) to E. The only part of the algorithm able to perform this operation is the subroutine *update*.[1] But when the subroutine adds the pair (s,t) to E then it must add all the pairs in $l(s,t)$. Hence, M' must also contain an ϵ-transition from p to q. As a consequence, if $w \in L(M')$ and $w \vdash w'$ then $w' \in L(M')$, i.e., $L(M')$ is closed under $\vdash$. Since the algorithm does not remove the original transitions from M, $L(M) \subseteq L(M')$ and, hence, the $\vdash$-closure of $L(M)$ is included in $L(M')$.

[1]Notice that the pair (s,t) can be added to E by the subroutine *update* either because it is the second argument in the call of the subroutine, or because it belongs to a list $l(p',q')$, where (p',q') is added to E in the same call, or because there is a path from s to t consisting of some arcs already in E and at least one arc added during the same call of *update*.

On the other hand, it can be easily shown that for each ϵ-transition of M' from a state p to a state q there exists an ϵ-reducible word z such that $q \in \delta(p, z)$ in M. Using this argument, from each word $w \in L(M')$ we can find a word $x \in L(M)$ such that $x \vdash^* w$. This permit us to conclude that $L(M')$ accepts the $\vdash$-closure of $L(M)$.

Now we show that the algorithm works in $O(n^4)$ time. A naive analysis gives a running time growing faster than n^4. The second for-loop iterates over all 4-tuples of states, and already gives us a total running time of n^4 multiplied by the running time of the work repeated with each iteration of the loop. Inside the loop the most expensive step is the subroutine *update*. This subroutine starts by adding an edge (p, q) to E. For each new edge (p', q') added to E the subroutine has to add all the edges in $l(p', q')$, while keeping the graph transitive. This seems to be an expensive part of the computation. However, we can observe that each set $l(p', q')$ contains less than n^2 elements. Furthermore, a set $l(p', q')$ is examined only once during the execution of the algorithm, namely when (p', q') is added to E. Hence, the total time spent while examining the sets l in all the calls of the subroutine *update* is $O(n^4)$. Furthermore, no more than n^2 edges can be inserted into G, and each insertion can be done in $O(n)$ amortized time while maintaining the transitive closure [26, 31]. Summing up, we get that the overall time of the algorithm is $O(n^4)$. □

By combining the results in Lemma 17 and Theorem 18, we are now able to show the following:

Proposition 19 *Given an ϵ-NFA $M = (Q, \Sigma, \delta, q_0, F)$ with n states, an ϵ-NFA M_r such that $L(M_r) = r(L(M))$ can be built in $O(n^4)$ time.*

Proof: The language $r(L(M))$ is the intersection of the $\vdash$-closure of $L(M)$ and $r(\Sigma^*)$. According to Theorem 18, from M we build an automaton M' accepting the $\vdash$-closure of $L(M)$. Hence, using standard constructions, from M' and the automaton obtained in Lemma 17 (whose size is fixed, if the input alphabet is fixed), we get the automaton M_r accepting $r(M) = L(M') \cap r(\Sigma^*)$. The most expensive part is the construction of M', which uses $O(n^4)$ time. □

The following result was known by Benois [5].

Corollary 20 *For any $L \subseteq \Sigma^*$, if L is regular then $r(L)$ is regular.*

This corollary does not hold for Thue systems in general. The next result from Shallit [3] proves that even if L is regular, it is possible for $r_g(L)$ to be noncontext-free, and hence also nonregular.

Proposition 21 *There exists a set of defining equations g and a regular language L such that $r_g(L)$ is not context-free.*

Proof: Let $\Sigma = \{a, b, c\}$, let $g = \{ab = ba, ac = ca, bc = cb\}$ and let $L = (abc)^*$. Then $r_g(L)$ is the language $\{x : |x|_a = |x|_b = |x|_c\}$, which is known to be noncontext-free. □

The analogue of Corollary 20 does not hold in the case of context-free languages either. The proof is supplied by Shallit [3]. For this we use the notion of a quotient of two languages.

Definition 22 *Given $L_1, L_2 \subseteq \Sigma^*$, the quotient of L_1 by L_2 is*

$$L_1/L_2 = \{w : \exists x \in L_2 \text{ such that } wx \in L_1\}$$

While the class of regular languages is closed under quotients, the class of context-free languages is not [20]. It turns out that the reduced representation of a language can be used to compute quotients.

Lemma 23 *For any two languages $L_1, L_2 \subseteq \Gamma^*$, the language $L_3 = r(L_1 L_2^{-1}) \cap \Gamma^*$ equals the quotient L_1/L_2.*

Proof: We notice that $r(wxx^{-1}) = w$, for each $w, x \in \Gamma^*$. Hence, given $w \in \Gamma^*$, it holds that $w \in L_3 = r(L_1 L_2^{-1}) \cap \Gamma^*$ if and only if there exists $x \in L_2$ such that $wx \in L_1$. Therefore $L_3 = L_1/L_2$. □

Corollary 24 *The class of context-free languages is not closed under $r()$.*

Proof: By contradiction, suppose that the class of context-free languages is closed under $r()$. Since this class is closed under the operations of reversal, morphism, concatenation and intersection with a regular language, for any two context-free languages L_1 and L_2

over Γ, the language $L_3 = r(L_1 L_2^{-1}) \cap \Gamma^*$ is also context-free. However, from Lemma 23, $L_3 = L_1/L_2$, implying that the class of context-free languages would be closed under quotient, a contradiction. □

Now we look at the set of equivalent words as the result of an operation on a language. Shallit [3] has shown that regular languages are not closed under this operation:

Proposition 25 *There exists a regular language L such that* $\mathrm{eq}(L)$ *is nonregular.*

Proof: Let $\Sigma = \{1, 1^{-1}\}$ and let $L = \{\epsilon\}$. Then $\mathrm{eq}(L)$ is the language $\{x \in \Sigma^* : |x|_1 = |x|_{-1}\}$, which is well known to be nonregular. □

As we see in the next proposition from Rampersad [3], it turns out that the set of equivalent words of a regular language is always context-free. More work on the relationships between various types of Thue systems and context-free languages has been done by Book [12].

Proposition 26 *Let L be a regular language over an alphabet Σ. The language* $\mathrm{eq}(L)$ *is context-free.*

Proof: Let M be a DFA accepting L. We first construct the ϵ-NFA M_r of Proposition 19 that accepts $r(L)$. We then reverse the transitions of M_r to obtain an ϵ-NFA A that accepts the reversal of $r(L)$. We now construct a PDA B that accepts $\mathrm{eq}(L)$. The operation of B is as follows. On input w, the PDA B reads each symbol of w and compares it with the symbol on top of the stack. If the symbol being read is a and the symbol on top of the stack is the inverse of a, the machine B pops the top symbol of the stack. Otherwise, the machine B pushes a on top of the stack. After the input w is consumed, the stack contains a word z that is equivalent to w. Moreover, since z does not contain any factor of the form aa^{-1}, the word z must equal $r(w)$ by Theorem 15. Finally, on ϵ-transitions, the PDA B pops each symbol of z off the stack and simulates the computation of A on each popped symbol. The net effect is to simulate A on z^R (the reversal of z). If z^R is accepted by A, the PDA B accepts w. Since z^R is accepted by A if and only if $z \in r(L)$, the PDA B accepts w if and only if $r(w) \in r(L)$. However, we have $r(w) \in r(L)$ if and only if $w \in \mathrm{eq}(L)$, so B accepts $\mathrm{eq}(L)$, as required. □

Next we resolve the question of whether context-free languages are closed under taking the set of equivalent words. The proof is from Shallit [37].

Proposition 27 *Let L be a context-free language over an alphabet Σ. The language* $\mathrm{eq}(L)$ *is not necessarily context-free.*

Proof: Suppose that we have a context-free language L such that $r(L)$ is not context-free. Now assume that $\mathrm{eq}(L)$ is context-free. From Lemma 17 we know that the language $r(\Sigma^*)$ is regular, so from the well-known fact that the intersection of a context-free language with a regular language is context-free, we have that $\mathrm{eq}(L) \cap r(\Sigma^*)$ is context-free. But $\mathrm{eq}(L) \cap r(\Sigma^*) = r(L)$, so we have a contradiction. □

3.2 State Complexity of Reduced Representations

By considering the state complexities of the automata involved in the construction used to prove Proposition 19, we can state the following upper bound on the state complexity of the reduced representation of a regular language in terms of its nondeterministic state complexity.

Proposition 28 *For any ϵ-NFA $M = (Q, \Gamma \cup \Gamma^{-1}, \delta, q_0, F)$ with n states, where $\Gamma = \{1, 2, \ldots, k\}$ and $\Gamma^{-1} = \{1^{-1}, 2^{-1}, \ldots, k^{-1}\}$ for some positive integer k, there exists a DFA of at most $2^n(2k+2)$ states that accepts $r(L(M))$.*

Proof: This upper bound follows from the algorithm in the proof of Theorem 18. The first part of the construction (i.e., the construction of the automaton M' accepting the $\vdash$-closure of the language accepted by M) does not increase the number of states. Using usual constructions, the resulting automaton M' can be converted into a DFA with 2^n states. Finally, to get an automaton accepting $r(L(M))$ we apply the cross-product construction to this automaton and to the DFA with $2k+2$ states accepting $r(\Sigma^*)$ obtained in Lemma 17. The intersection results in a DFA of no more than $2^n(2k+2)$ states. □

Pighizzini [30] has supplied the lower bounds that follow, but first we give the Myhill-Nerode theorem, which is required for one of the proofs. A proof for the Myhill-Nerode theorem can be found in many texts including the one by Hopcroft and Ullman [25].

Theorem 29 *For a language $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$, let R_L be the equivalence relation defined by: xR_Ly if and only if for all $z \in \Sigma^*$, we have $xz \in L$ exactly when $yz \in L$. Then the following statements are equivalent.*

1. *L is regular.*
2. *L is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.*
3. *R_L is of finite index.*

A well-known consequence of the Myhill-Nerode theorem is the following corollary.

Corollary 30 *The state complexity of a language L corresponds to the index of R_L, the relation defined in the Myhill-Nerode theorem.*

Now we give the bounds from Pighizzini [30].

Proposition 31 *Let k be a positive integer and define $\Gamma_k = \{1, 2, \ldots, k\}$ and $\Gamma_k^{-1} = \{1^{-1}, 2^{-1}, \ldots, k^{-1}\}$.*

1. *For each pair of integers $k \geq 2$ and $n \geq 1$, there exists an n-state NFA $M_{k,n}$ over $\Gamma_k \cup \Gamma_k^{-1}$ such that the smallest DFA accepting $r(L(M_{k,n}))$ has 2^n states.*
2. *For each pair of integers $k \geq 3$ and $n \geq 1$, there exists a DFA $M_{k,n}$ over $\Gamma_k \cup \Gamma_k^{-1}$ of $3n$ states such that the smallest DFA accepting $r(L(M_{k,n}))$ has 2^n states.*

Proof:

1. Let k be any integer ≥ 2. Since any language defined over just Γ_k is exactly the same as its reduced representation, as long as there exists an n-state NFA $M_{k,n}$ for each integer n such that $\mathrm{sc}(L(M_{k,n})) = 2^n$, then we have the desired bound. There do indeed exist such NFAs as shown, for example, by Moore [28].

2. If we let $L_n = (1+2)^*1(33^{-1}(1+2))^{n-1}$, we get $r(L_n) = (1+2)^*1(1+2)^{n-1}$. It is easy to see that L_n is accepted by a DFA with $3n$ states (see Figure 3.2). Using Corollary 30 we can show that any DFA accepting $r(L_n)$ requires 2^n states. Let x, y be two distinct elements of $(1+2)^n$. Then there is a position j, $1 \leq j \leq n$, at which x and y have a different letter. Without loss of generality, assume x has a 1 at position j. Then $x2^{j-1} \in r(L_n)$, but $y2^{j-1} \notin r(L_n)$. So each of the 2^n strings in the language $(1+2)^n$ are in different equivalence classes by the equivalence relation $R_{r(L_n)}$, and by Corollary 30, any DFA accepting $r(L_n)$ requires at least 2^n states.

□

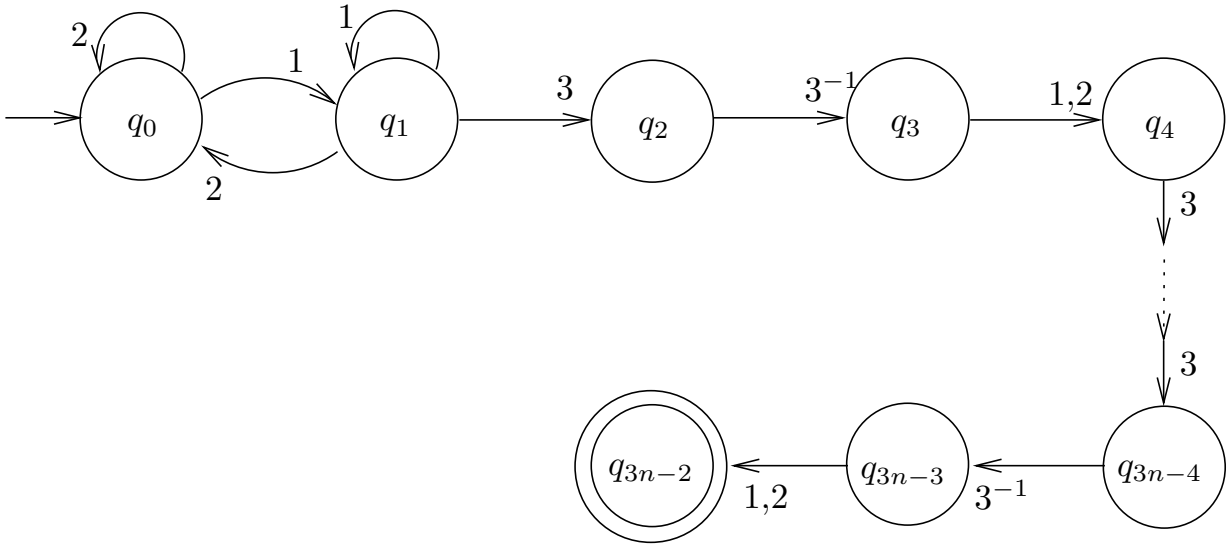

Figure 3.2: A DFA of $3n$ states that accepts L_n. The dead state is not shown.

While we know that the worst-case state complexity of the reduced representation is exponential, no tight bound is known so we have the following open problem:

Open Problem 32 *What is the greatest integer function $f(n)$ such that for any integer n we can find a language L_n such that $\mathrm{sc}(L_n) = n$ and $\mathrm{sc}(r(L_n)) \geq f(n)$?*

For the nondeterministic case, Proposition 28 and the first part of Proposition 31, give us the following corollary.

Corollary 33 *Let $k \geq 1$ be an integer, $\Gamma_k = \{1, 2, \ldots, k\}$, and $\Gamma_k^{-1} = \{1^{-1}, 2^{-1}, \ldots, k^{-1}\}$. For any ϵ-NFA, M with n states, over the alphabet $\Gamma_k \cup \Gamma_k^{-1}$, there exists an NFA of at most $n(2k+2)$ states that accepts $r(L(M))$, and for each pair of integers $k \geq 2$ and $n \geq 1$ there exists an n-state NFA $M_{k,n}$ over $\Gamma_k \cup \Gamma_k^{-1}$ such that the smallest NFA accepting $r(L(M_{k,n}))$ has n states.*

We also have the following open problem.

Open Problem 34 *What is the greatest integer function $f(n)$ such that for any integer n we can find a language L_n such that $\mathrm{nsc}(L_n) = n$ and $\mathrm{nsc}(r(L_n)) \geq f(n)$?*

3.3 Shortest ϵ-Reducible Words

In this section we turn our attention to studying bounds on the length of shortest ϵ-reducible strings in regular languages. Formally, we ask: given an NFA M of n states such that $\epsilon \in r(L(M))$, what is the shortest $w \in L(M)$ such that $r(w) = \epsilon$? This is actually the equivalent of asking for the rational index of the language $\mathrm{eq}(\{\epsilon\})$. For a language L, its rational index ρ_L is a function defined by Boasson et al. [11] as follows in our own words.

Definition 35 *Let R represent the set of regular languages, let n be a positive integer, and let L be a language. Then*

$$\rho_L(n) = \max\{\min\{|x| : x \in L \cap K, L \cap K \neq \emptyset, K \in R, \mathrm{nsc}(K) = n\}\}.$$

Recall from Proposition 26 that $\mathrm{eq}(\{\epsilon\})$ is a context-free language. The more general problem of finding bounds for the rational index of context-free languages was studied by Boasson et al. [11], and the upper and lower bounds of 2^{pn^2}, where p is a function of the size of the context-free grammar that generates the context-free language, and $2^n - 2$, respectively, were found. For our special case we provide upper and lower bounds of 2^{n^2-n} and 2^{n-2}, respectively. We begin with the upper bound from Rampersad [3].

Theorem 36 *For any NFA $M = (Q, \Sigma, \delta, q_0, F)$ with n states such that there exists $w \in L(M)$ with $r(w) = \epsilon$, there exists $w' \in L(M)$ such that $|w'| \leq 2^{n^2-n}$ and $r(w') = \epsilon$.*

Proof: Suppose M accepts $w \in \Sigma^+$ such that $r(w) = \epsilon$. Then w can be decomposed in at least one of two ways. Either there exist $u, v \in \Sigma^+$ such that $w = uv, r(u) = \epsilon$ and $r(v) = \epsilon$ (Case 1), or there exist $u \in \Sigma^*, a \in \Sigma$ such that $w = aua^{-1}$ and $r(u) = \epsilon$ (Case 2). Any factor z of w such that $r(z) = \epsilon$ can also be decomposed in at least one of these two ways, so we can recursively decompose w and the resulting factors until we have decomposed w into single symbols. It follows that we can specify a certain type of parse tree such that M accepts $w \in \Sigma^*$ with $r(w) = \epsilon$ if and only if we can build this type of parse tree for w.

Define our parse tree for a given w as follows. Every internal node corresponds to a factor z of w such that $r(z) = \epsilon$, and the root of the whole tree corresponds to w. The leaves store individual symbols. When read from left to right, the symbols in the leaves of any subtree form the word that corresponds to the root of the subtree. Each internal node is of one of two types:

1. The node has two children, both of which are internal nodes that serve as roots of subtrees (corresponds to Case 1).

2. The node has three children, where the left and the right children are single symbols that are inverses of each other, and the child in the middle is empty or it is an internal node that is the root of another subtree (corresponds to Case 2).

An example is shown in Figure 3.3. Now, we fix an accepting computation of M on input w. We label each internal node t with a pair of states $p, q \in Q$ such that if z is the factor of w that corresponds to the subtree rooted at t, and $w = xzy$, then $p \in \delta(q_0, x)$ and $q \in \delta(p, z)$ are the states reached after reading the input prefixes x and xz, respectively, during the accepting computation under consideration. (This also implies that $q_f \in \delta(q, y)$, with $q_f \in F$, and (q_0, q_f) is the label associated with the root of the tree.)

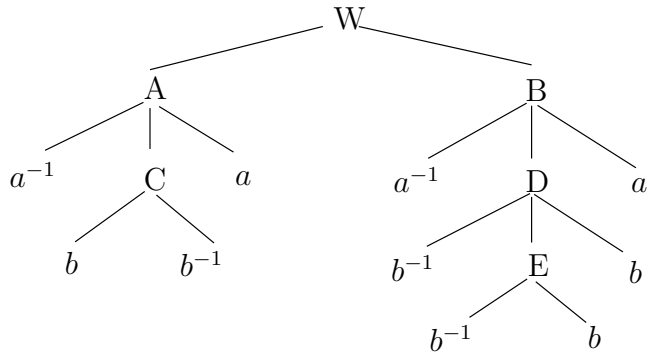

Figure 3.3: A sample parse tree for the word $w = a^{-1}bb^{-1}aa^{-1}b^{-1}b^{-1}bba$, without the state pair labels.

If the parse tree of w has two nodes t and u with the same state-pair label such that u is a descendent of t, then there exists a word shorter than w that is accepted and reduces to the empty word. This is because we can replace the subtree rooted at t with the subtree rooted at u. Furthermore, if an internal node t is labeled with a pair (q, q), for some $q \in Q$, then the factor z corresponding to the subtree rooted at t can be removed from w, obtaining a shorter ϵ-reducible word. Hence, by a pigeonhole argument, we conclude that the height of the subtree corresponding to the shortest ϵ-reducible word w' is at most $n^2 - n$. We now observe that the number of leaves of a parse tree of height k defined according to our rules is at most 2^k. Such a tree is given by the complete binary tree of height k, which has no nodes with three children. The avoidance of nodes with three children is important because such nodes fail to maximize the number of internal nodes in the tree, which in turn results in less than the maximum number of leaves. This permits us to conclude that $|w'| \leq 2^{n^2-n}$. □

Now we show that there is a lower bound that is exponential in the alphabet size and nondeterministic state complexity We show this by giving a bound that is stronger in the sense that it holds even in the deterministic case.

Theorem 37 *For each integer $n \geq 3$ there exists a DFA M_n with $n + 1$ states over the alphabet $\Sigma = \Gamma \cup \Gamma^{-1}$, where $\Gamma = \{1, 2, \ldots, n-2\}$ and $\Gamma^{-1} = \{1^{-1}, 2^{-1}, \ldots, (n-2)^{-1}\}$, with the property that if $w \in L(M_n)$ and $r(w) = \epsilon$, then $|w| \geq 2^{n-1}$.*

Proof: The proof is constructive. Let M_n be the DFA $(Q, \Sigma, \delta, q_0, F)$, illustrated in Figure 3.4, where $Q = \{q_{-1}, q_0, q_1, \ldots, q_{n-1}\}$, $F = \{q_1\}$, and

$$\delta(q_a, c) = \begin{cases} q_1, & \text{if } c = 1 \text{ and } a = 0; \\ q_{a+1}, & \text{if } c = a^{-1} \text{ and } 1 \leq a \leq n-2; \\ q_0, & \text{if either } \quad c = a \text{ and } 1 \leq a \leq n-2, \\ & \qquad \text{or } \quad c = 1^{-1} \text{ and } a = n-1. \end{cases}$$

Any other transitions lead to the dead state q_{-1}.

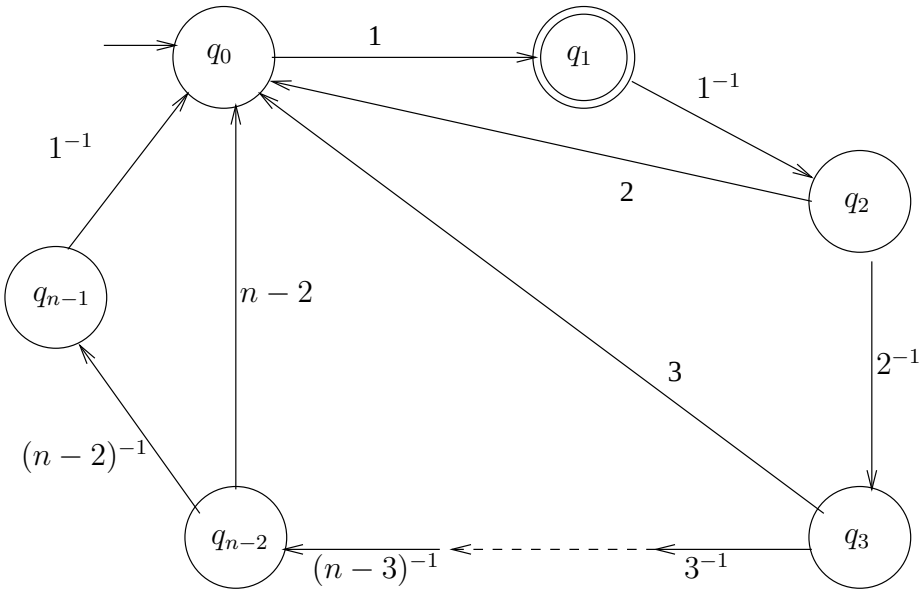

Figure 3.4: M_n: an $n+1$ state DFA that has a shortest ϵ-reducible word of length $\geq 2^{n-1}$. The dead state is not shown.

Now we show that M_n has the desired property. Assume there exists $w \in L(M_n)$ such that $r(w) = \epsilon$. Then $|w|_a = |w|_{a^{-1}}$ for all $a \in \Sigma$. Since all words in $L(M_n)$ must contain the symbol 1 (due to the single incoming transition to the only accepting state), it follows that w must also contain 1^{-1}. Furthermore, the only possible transition from q_1 not leading to the dead state uses the symbol 1^{-1}. Hence w must begin with the prefix 11^{-1}. Since $\delta(q_0, 11^{-1}) = q_2$ and the only two transitions that leave q_2 are on 2 and 2^{-1}, w must contain both of 2 and 2^{-1}. Now assume that w contains the symbol i^{-1} with $1 < i < n-2$. Then

the state q_{i+1} must be reached while reading w, thus implying that the symbols $(i+1)$ and $(i+1)^{-1}$ also appear in w. Therefore, by induction, w must contain at least one occurrence of each $a \in \Sigma$.

Now we claim that w must contain at least 2^{n-2-a} occurrences of the symbol a, for $1 \leq a \leq n-2$, and hence at least 2^{n-2-a} occurrences of the symbol a^{-1}.

We prove the claim by induction on $n-2-a$. The basis, $a = n-2$, follows from the earlier argument that w must contain at least one occurrence of each $a \in \Sigma$. Now, suppose the claim holds for $k = n-2-a$. We prove it is true for $k+1 = n-2-(a-1)$. By the induction hypothesis, w contains at least 2^k occurrences of the symbol a and at least 2^k occurrences of a^{-1}. Observing the structure of the automaton, we conclude that to have such a number of occurrences of the two letters a and a^{-1}, the state q_a must be visited at least 2^{k+1} times. On the other hand, the only transition entering q_a is from the state q_{a-1} on the letter $(a-1)^{-1}$. Hence, w must contain at least $2^{k+1} = 2^{n-2-(a-1)}$ occurrences of $(a-1)^{-1}$ and also at least $2^{k+1} = 2^{n-2-(a-1)}$ occurrences of $a-1$. This proves the claim.

By computing the sum over all alphabet symbols, we get that $|w| \geq 2(2^{n-2}-1)$. However, since the symbol $(n-2)^{-1}$ must always be followed by the symbol 1^{-1}, w must actually contain one additional occurrence of each of 1 and 1^{-1}. Thus $|w| \geq 2^{n-1}$. □

It turns out that w_n, the shortest ϵ-reducible word accepted by the DFA M_n in the proof of Theorem 37, is related to the well-known *ruler sequence*, $(\nu_2(n))_{n\geq 1}$, where $\nu_2(n)$ denotes the exponent of the highest power of 2 dividing n. This sequence has many interesting characterizations including being the lexicographically least infinite squarefree word over $\mathbb{Z}$. We make the following definition.

Definition 38 *For all integers $k > 0$, let $r_k = (\nu_2(n)_{1\leq n<2^k})$ be the prefix of length $2^k - 1$ of the ruler sequence.*

For example, $r_3 = 0102010$. Observe that in general we have the recursive definition $r_{k+1} = r_k k r_k$.

Proposition 39 *Define the homomorphism $h : (\Gamma \cup \Gamma^{-1})^* \to (\Gamma \cup \{0\})^*$ such that $h(a) = a-1$ for $a \in \Gamma$, and $h(a) = \epsilon$ for $a \in \Gamma^{-1}$. Then for all integers $n \geq 3$, we have $h(w_n) = r_{n-2}0$.*

Proof: Let w'_n be the prefix of w of length $|w| - 2$. This is well defined for $n \geq 3$. Also let $w_n = w'_n = \epsilon$ for $n = 2$. Then observe that for any integer $n \geq 3$, we have $w_n = w'_{n-1}(n-2)w'_{n-1}(n-2)^{-1}1^{-1}1$ (one can verify that this word is indeed accepted by M_n, and that it is of the minimal length specified in Theorem 37). Now we prove by induction that $h(w_n) = r_{n-2}0$. For the base case, observe that $h(w_3) = 00$ and $r_{n-2} = 0$. Now assume the inductive hypothesis that $h(w_k) = r_{k-2}0$ for some integer $k \geq 3$. It follows from this assumption and from the structure of w_k that $h(w'_k) = r_{k-2}$. From the definition of h we have $h(w_{k+1}) = h(w'_k)(k-2)h(w'_k)1$. From our inductive hypothesis we have $h(w_{k+1}) = r_{k-2}(k-2)r_{k-2}1$. Since $r_{k-1} = r_{k-2}(k-2)r_{k-2}$, our result follows from induction. □

While Theorems 36 and 37 show us that the length of the shortest ϵ-reducible word can be exponential in worst-case, no tight bound is known so we have the following open problem:

Open Problem 40 *What is the greatest integer function $f(n)$ such that for any integer n we can find a language L_n such that* $\mathrm{nsc}(L_n) = n$ *and the length of the shortest ϵ-reducible word is $\geq f(n)$?*

3.4 Bounds for Restricted Alphabets

We now turn our attention to shortest ϵ-reducible strings over restricted alphabets. The next theorem from Pighizzini [3] shows that over a fixed alphabet size we can still get an exponential lower bound in terms of state complexity.

Theorem 41 *For each integer $n \geq 1$ there exists a DFA, M_n with $3n + 1$ states over the alphabet $\Sigma = \Gamma \cup \Gamma^{-1}$, where $\Gamma = \{1, 2\}$ and $\Gamma^{-1} = \{1^{-1}, 2^{-1}\}$, with the property that the only word $w \in L(M_n)$ such that $r(w) = \epsilon$ has length $|w| = 3 \cdot 2^n - 4$.*

Proof: The proof is constructive. Let M_n be the DFA $(Q, \Sigma, \delta, q_n, F)$ where $Q = \{q_{-1}, q_0, q_1, p_1\} \cup \{p_i, q_i, r_i : 2 \leq i \leq n\}$, $F = \{p_n\}$, and

$$\delta(q_a, c) = q_{a-1}, \quad \text{if } 1 \leq a \leq n, \text{ and either } \quad c = 1 \text{ and } a \equiv 1 \pmod 2,$$
$$\text{or} \quad c = 2 \text{ and } a \equiv 0 \pmod 2;$$

$$\delta(p_a, c) = \begin{cases} p_{a+1}, & \text{if } 1 \le a \le n-1, \text{ and either } \quad c = 1 \text{ and } a \equiv 0 \pmod 2, \\ & \qquad\qquad \text{or } \quad c = 2 \text{ and } a \equiv 1 \pmod 2; \\ r_{a+1}, & \text{if } 1 \le a \le n-1, \text{ and either } \quad c = 1^{-1} \text{ and } a \equiv 0 \pmod 2, \\ & \qquad\qquad \text{or } \quad c = 2^{-1} \text{ and } a \equiv 1 \pmod 2; \end{cases}$$

$$\delta(r_a, c) = q_{a-1}, \quad \text{if } 2 \le a \le n, \text{ and either } \quad c = 1^{-1} \text{ and } a \equiv 1 \pmod 2, \\ \text{or } \quad c = 2^{-1} \text{ and } a \equiv 0 \pmod 2;$$

$$\delta(q_0, a^{-1}) = p_1.$$

Any other transitions lead to the dead state q_{-1}. An example for $n = 4$ is illustrated in Figure 3.5.

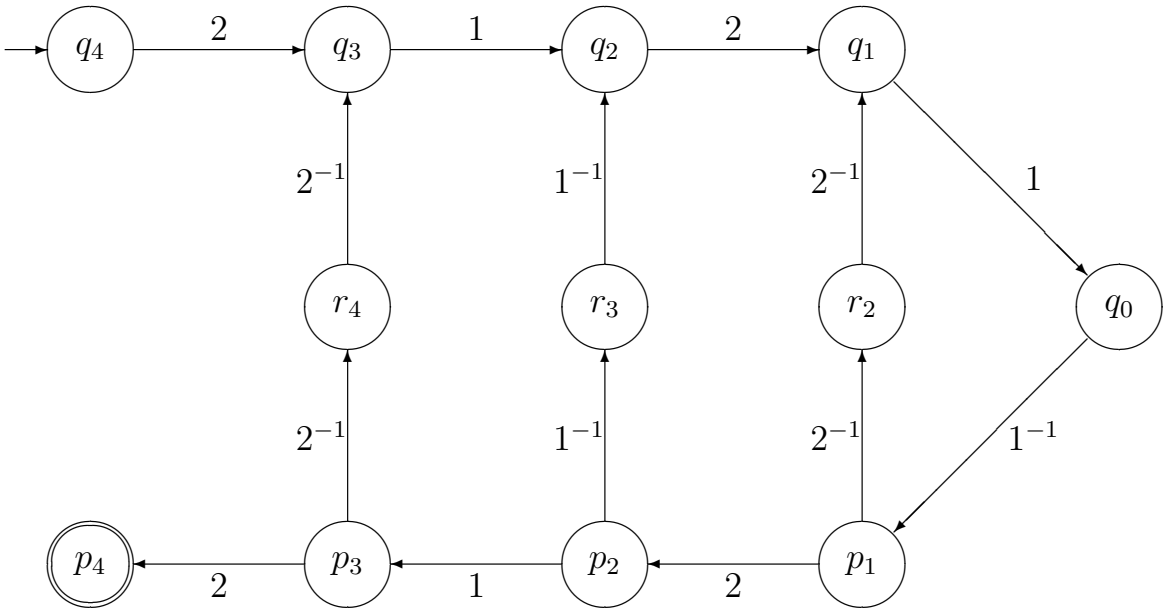

Figure 3.5: M_4: a $3 \cdot 4 + 1$ state DFA with the property that the only ϵ-reducible word accepted by it has length $3 \cdot 2^4 - 4$. The dead state is not shown.

In order to prove the statement, for each integer $m \geq 0$, let us consider the set C_m of pairs of states that exclude the dead state, which are connected by an ϵ-reducible word of length m, i.e.

$$C_m = \{(s', s'') \in Q' \times Q' : \exists w \in \Sigma^* \text{ s.t. } |w| = m, r(w) = \epsilon, \text{ and } \delta(s', w) = s''\},$$

where $Q' = Q \setminus \{q_{-1}\}$. We notice that $C_m = \emptyset$, for m odd. Furthermore $C_0 = \{(s, s) : s \in Q'\}$ and, for $m > 0$, $C_m = C'_m \cup C''_m$, where:

$$\begin{aligned} C'_m = \{(s', s'') \quad : \quad & \exists (r', r'') \in C_{m-2}, a \in \Sigma \\ & \text{s.t. } \delta(s', a) = r' \text{ and } \delta(r'', a^{-1}) = s''\}, \end{aligned} \tag{3.1}$$

$$\begin{aligned} C''_m = \{(s', s'') \quad : \quad & \exists m', m'' > 0, (s', r') \in C_{m'}, (r'', s'') \in C_{m''} \\ & \text{s.t. } m' + m'' = m \text{ and } r' = r''\}. \end{aligned} \tag{3.2}$$

We claim that, for each $m \geq 1$:

$$C_m = \begin{cases} \{(q_k, p_k)\}, & \text{if } \exists k,\ 1 \leq k \leq n, \text{ s.t. } m = 3 \cdot 2^k - 4; \\ \{(q_k, r_k), (r_k, p_k)\}, & \text{if } \exists k,\ 2 \leq k \leq n, \text{ s.t. } m = 3 \cdot 2^{k-1} - 2; \\ \emptyset, & \text{otherwise.} \end{cases} \tag{3.3}$$

We prove (3.3) by induction on m.

As already noticed, $C_1 = \emptyset$. By inspecting the transition function of M_n, we can observe that $C_2 = \{(q_1, p_1)\}$. Notice that $2 = 3 \cdot 2^1 - 4$. This proves the basis.

For the inductive step, we now consider $m > 2$ and we suppose Equation (3.3) true for integers less than m.

First, we show that we can simplify the formula in Equation (3.2) for C''_m. In fact, using the inductive hypothesis, for $0 < m', m'' < m$, the only possible $(s', r') \in C_{m'}$ and $(r'', s'') \in C_{m''}$ satisfying $r' = r''$ are the pairs $(q_j, r_j), (r_j, p_j) \in C_{3 \cdot 2^{j-1} - 2}$, obtained by taking $m' = m'' = 3 \cdot 2^{j-1} - 2$, for suitable values of j. This, together with the condition $m' + m'' = m$, restricts the set C''_m to:

$$C''_m = \{(s', s'') \mid \exists r \in Q : (s', r) \in C_{m/2} \text{ and } (r, s'') \in C_{m/2}\}. \tag{3.4}$$

Now we consider three subcases:

Case 1: $m = 3 \cdot 2^k - 4$, with $k \geq 2$.
An easy verification shows that $m-2$ cannot be expressed in the form $3 \cdot 2^j - 4$ or in the form $3 \cdot 2^j - 2$, for any j. Hence, by the inductive hypothesis, $C_{m-2} = \emptyset$. By Equation (3.1), this implies $C'_m = \emptyset$, and then $C_m = C''_m$.

We now compute C''_m using Equation (3.4) and the set $C_{m/2}$ obtained according to the inductive hypothesis. We observe that $m/2 = 3 \cdot 2^{k-1} - 2$. Hence, for $k \leq n$, $C_{m/2} = \{(q_k, r_k), (r_k, p_k)\}$ and, thus, $C_m = C''_m = \{(q_k, p_k)\}$. On the other hand, if $k > n$ then $C_{m/2} = \emptyset$, which implies $C_m = C''_m = \emptyset$.

Case 2: $m = 3 \cdot 2^{k-1} - 2$, with $k \geq 2$.
First, we observe that $m/2$ cannot be written either as $3 \cdot 2^j - 4$ or as $3 \cdot 2^j - 2$. Hence, by the inductive hypothesis, the set C''_m must be empty. Thus, $C_m = C'_m$.

We compute C'_m as in Equation (3.1), using the set C_{m-2} obtained according to the induction hypothesis. We notice that $m - 2 = 3 \cdot 2^{k-1} - 4$. If $k > n+1$ then $C_{m-2} = \emptyset$, thus implying $C_m = C'_m = \emptyset$. Otherwise, $C_{m-2} = \{(q_{k-1}, p_{k-1})\}$. In order to obtain all the elements of C'_m, we have to examine the transitions entering q_{k-1} or leaving p_{k-1}. For $k = n+1$ there are no such transitions and, hence, $C_m = C'_m = \emptyset$. For $k \leq n$ all the transitions entering q_{k-1} or leaving p_{k-1} involve the same symbol $a \in \{1, 2\}$ or its inverse: there are exactly two transitions entering q_{k-1} ($\delta(q_k, a) = \delta(r_k, a^{-1}) = q_{k-1}$) and exactly two transitions leaving p_{k-1} ($\delta(p_{k-1}, a) = p_k$ and $\delta(p_{k-1}, a^{-1}) = r_k$). Hence, by the appropriate combinations of these transitions with the only pair (q_{k-1}, p_{k-1}) in C_{m-2}, we get that $C_m = C'_m = \{(q_k, r_k), (r_k, p_k)\}$.

Case 3: Remaining values of m.
If $m = 3 \cdot 2^{k-1}$ with $k \geq 2$, then $C_{m-2} = \{(q_k, r_k), (r_k, p_k)\}$. All the transitions entering or leaving r_k use the same symbol a^{-1}, with $a \in \{1, 2\}$, while all the transitions entering q_k or leaving p_k use the other symbol $b \in \{1, 2\}$, $b \neq a$, or b^{-1}. Hence, from Equation (3.1), $C'_m = \emptyset$.

For all the other values of m, the form of $m-2$ is neither $3 \cdot 2^j - 4$ nor $3 \cdot 2^j - 2$. This implies that $C_{m-2} = \emptyset$ and, then, C'_m must be empty. Hence, we conclude that $C_m = C''_m$.

Suppose $C''_m \neq \emptyset$. From Equation (3.4) and the inductive hypothesis, it follows that $m/2 = 3 \cdot 2^{j-1} - 2$ for some j, thus implying $m = 3 \cdot 2^j - 4$. This is a contradiction, because the values of m we are considering are not of this form. Hence, $C_m = C''_m$ must be empty.

This completes the proof of Equation (3.3).

Recall that the initial state of M_n is q_n, while the only final state is p_n. Hence, the length of the shortest ϵ-reducible word accepted by M_n is the smallest integer m such that $(q_n, p_n) \in C_m$. According to Equation (3.3), we conclude that such a length is $3 \cdot 2^n - 4$.

From Equation (3.3), it also follows that there are no ϵ-reducible words accepted by M_n with length $\neq 3 \cdot 2^n - 4$. With some small refinements in the argument used to prove Equation (3.3), we can show that M_n accepts exactly one ϵ-reducible word. In particular, for $k \geq 1$ we consider:

$$w_k = \begin{cases} 11^{-1}, & \text{if } k = 1; \\ 1w_{k-1}1^{-1}1^{-1}w_{k-1}1, & \text{if } k > 1 \text{ and } k \text{ odd}; \\ 2w_{k-1}2^{-1}2^{-1}w_{k-1}2, & \text{otherwise.} \end{cases}$$

By an inductive argument it can be proved that w_k is the only ϵ-reducible word such that $\delta(q_k, w_k) = p_k$ and $|w_k| = 3 \cdot 2^k - 4$. □

Restricting our alphabet further to the special case where $\Sigma = \{1, 1^{-1}\}$, we give a cubic upper bound and quadratic lower bound. The next result from Rampersad [3] gives the upper bound. Boasson et al. [11] use a very similar proof to show that the language of balanced parentheses has a cubic upper bound for its rational index.

Theorem 42 *Let $M = (Q, \{1, 1^{-1}\}, \delta, q_0, F)$ be an NFA with n states such that $\epsilon \in r(L(M))$. Then M accepts an ϵ-reducible word of length at most $n(2n^2 + 1)$.*

Proof: We prove the result by contradiction. Assume the shortest $w \in L(M)$ such that $r(w) = \epsilon$ has $|w| > n(2n^2 + 1)$. For $z \in \Sigma^*$ let b refer to the function on words $b(z) = |z|_1 - |z|_{1^{-1}}$. Roughly speaking, the function b measures the "balance" between the number of occurrences of the symbol 1 and those of the symbol 1^{-1} in a word.

Suppose that no factor w' of w has $|b(w')| > n^2$. Then the function b can take on at most $2n^2 + 1$ distinct values. Since $|w| > n(2n^2 + 1)$, there must be a value C such that b takes the value C for more than n different prefixes of w. That is, there is some $\ell \geq n$ such that $w = xy_1y_2 \cdots y_\ell z$ where the y_i are nonempty and

$$b(x) = b(xy_1) = b(xy_1y_2) = \cdots = b(xy_1y_2 \cdots y_\ell).$$

Consider a sequence of $\ell+1$ states $p_0, p_1, \ldots, p_\ell \in Q$, and a state $q_f \in F$, such that during an accepting computation M makes the following transitions:

$$p_0 \in \delta(q_0, x), p_1 \in \delta(p_0, y_1), \ldots, p_\ell \in \delta(p_{\ell-1}, y_\ell), q_f \in \delta(p_\ell, z)$$

Since $\ell \geq n$, a state must repeat in the above sequence, say $p_i = p_j$ with $i < j$. Then $u = xy_1 \cdots y_i y_{j+1} \cdots y_\ell z$ is shorter than w (since we have omitted $y_{i+1} \cdots y_j$) and it is accepted by M. Furthermore, observing that $b(y_{i+1} \cdots y_j) = 0$, we conclude that $r(u) = \epsilon$. Since $|u| < |w|$, this is a contradiction to our choice of w. Hence, w must contain a factor w' such that $|b(w')| > n^2$.

Let y be the shortest factor of w such that $b(y) = 0$ and $|b(y')| > n^2$ for some prefix y' of y. We can write $w = xyz$, for suitable words x, z. Let D be the maximum value of $|b(y')|$ over all prefixes y' of y. For $i = 0, 1, 2, \ldots, D$, let $R(i)$ be the shortest prefix of y with $b(R(i)) = i$. Similarly, let $S(i)$ be the longest prefix of y with $b(S(i)) = i$. Again, consider an accepting computation of M on input w. For each pair $[R(i), S(i)]$, let $[P(i), Q(i)]$ be the pair of states such that M is in state $P(i)$ after reading $xR(i)$, and M is in state $Q(i)$ after reading $xS(i)$. Since $D > n^2$, some pair of states repeats in the sequence $\{[P(i), Q(i)]\}$. That is, there exists $j < k$ such that $[P(j), Q(j)] = [P(k), Q(k)]$. We may therefore omit the portion of the computation that occurs between the end of $R(j)$ and the end of $R(k)$ as well as that which occurs between the end of $S(k)$ and the end of $S(j)$ to obtain a computation accepting a shorter word u such that $r(u) = \epsilon$. Again we have a contradiction, and our result follows. □

The following result from Pighizzini [3] gives a quadratic lower bound, which holds even in the deterministic case.

Theorem 43 *For each integer $n \geq 0$ there exists a DFA M_n with $n+1$ states, over the alphabet $\Sigma = \{1, 1^{-1}\}$, such that the only ϵ-reducible word $w \in L(M_n)$, has length $(n^2-1)/2$ if n is odd, and $n^2/2$ if n is even.*

Proof: The proof is constructive. Let M_n be the DFA $(Q, \Sigma, \delta, q_0, F)$, illustrated in Figure 3.6, where $Q = \{q_{-1}\} \cup \{q_i : 0 \leq i < n\}$, $F = \{q_{\lfloor \frac{n}{2} \rfloor}\}$, and

$$\delta(q_a, c) = \begin{cases} q_{a+1}, & \text{if either } c = 1 \text{ and } 0 \leq a < \lfloor \frac{n}{2} \rfloor, \\ & \text{or } c = 1^{-1} \text{ and } \lfloor \frac{n}{2} \rfloor \leq a \leq n-2; \\ q_{a \bmod 2}, & \text{if } c = 1^{-1} \text{ and } a = n-1. \end{cases}$$

Any other transitions lead to the dead state, q_{-1}.

Observe that if n is odd, then each word w accepted by M_n has the form $w = (1^{\frac{n-1}{2}}1^{-\frac{n+1}{2}})^{\alpha}1^{\frac{n-1}{2}}$, for an $\alpha \geq 0$. Computing the "balance" function b introduced in the proof of Theorem 42, we get $b(w) = \frac{1}{2}(n-1-2\alpha)$, which is 0 if and only if $\alpha = \frac{n-1}{2}$. Finally, by computing the length of w for such an α, we obtain $(n+1)(n-1)/2 = (n^2-1)/2$.

Similarly, in the case of n even, we can prove that the only ϵ-reducible word accepted by M_n has length $n^2/2$. □

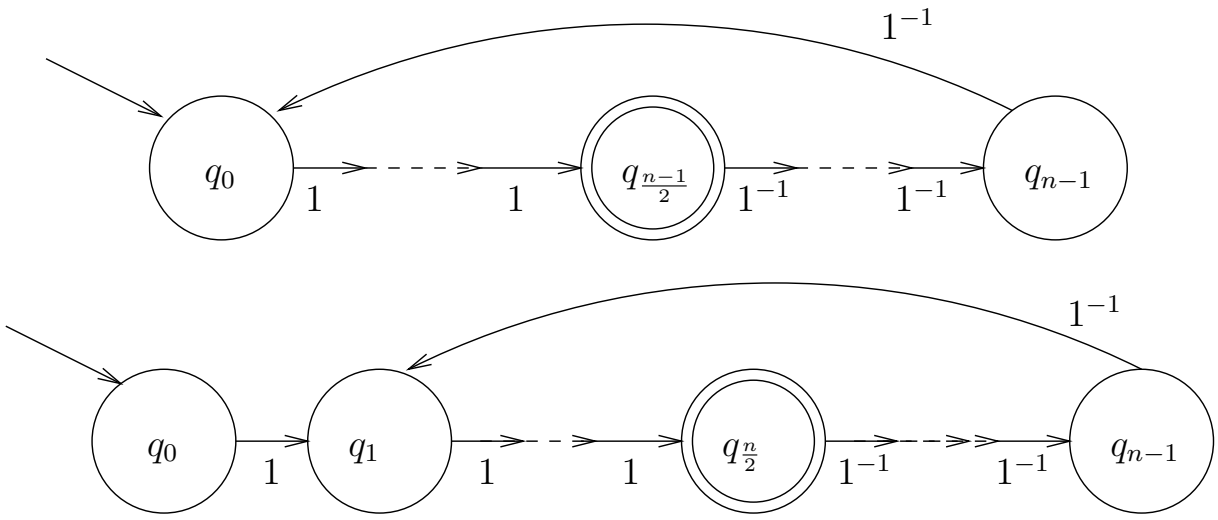

Figure 3.6: Top: M_n where n is odd. Bottom: M_n where n is even.

We now know that in the special case where $\Sigma = \{1, 1^{-1}\}$ the length of the shortest ϵ-reducible word is polynomial in worst-case, but no tight bound is known so we have the following open problem:

Open Problem 44 *What is the greatest integer function $f(n)$ such that for any integer n we can find a language L_n over $\Sigma = \{1, 1^{-1}\}$ such that* $\mathrm{nsc}(L_n) = n$ *and the length of the shortest ϵ-reducible word is $\geq f(n)$?*

Chapter 4

Open Problems

Open problems were previously stated regarding the shortest string in the intersection of two positive closures of regular languages (Open Problem 12), the shortest string in the intersection of three or more regular languages (Open Problem 14), tight bounds for the state complexity and nondeterministic state complexity of reduced representations (Open Problems 32 and 34 and tight bounds for the length of shortest ϵ-reducible strings (Open Problems 40 and 44). We now discuss some additional problems related to the concept of shortest strings that were studied but not solved.

Open Problem 45 *Given a DFA M, is there an efficient method for determining if $L(M)$ contains a squarefree word?*

A word $w \in \Sigma^*$ is squarefree if it contains no factor of the form $u = xx$, where $u, x \in \Sigma^*$. Squarefree words have been the subject of much study over the last century and Berstel has written surveys of results that go back as far as Thue [8, 9]. Open Problem 45 is very similar to those discussed by Anderson et al. [1].

The brute force solution to deciding if a language contains a squarefree word is to check for squares in all words accepted by the language starting the the shortest, and to terminate if we find a word that contains no squares. In cases where there are no such words, we need an upper bound on the length of the shortest squarefree word accepted by regular languages to know when to terminate. Hence, to determine the complexity of this brute force method we need good bounds on this length. This leads to the following open shortest string problem.

Open Problem 46 *What are good bounds on the length of the shortest squarefree word accepted by a DFA?*

If we let L' be the language of all squarefree words, then the length of the shortest squarefree word in a language L can be expressed at $\text{lss}(L \cap L')$. Unfortunately, L' is not even context-free [8], let alone regular, so we cannot apply our results from Chapter 2. While the author has not solved this problem, in attempting to construct an example that gives a good lower bound, a sequence of automata that relates the ruler sequence and Pascal's triangle was found.

Recall from Section 3.3 that the ruler sequence is defined by $(\nu_2(n))_{n\geq 1}$, where $\nu_2(n)$ denotes the exponent of the highest power of 2 dividing n. Also recall the sequence of prefixes of the ruler sequence from Definition 38. Then we define our sequence of DFAs, which we call the *ruler automata* as follows. For each $n \geq 1$, the ruler automaton, $R(n)$, is the DFA $(Q_n, \Sigma_n, q_0, F_n, \delta_n)$, where $Q_n = \{q_0, q_1, q_2..., q_{n-1}\}$, $\Sigma_n = \{0, 1, 2, ..., n-2\}$, $F_n = q_{n-1}$, and for $q_a \in Q, b \in \Sigma$, $\delta_n(q_a, b) = q_{a-b+1}$ when $a - b + 1 > 0$, and $\delta_n(q_a, b) = q_0$ otherwise. An example of this construction is shown in Figure 4.1.

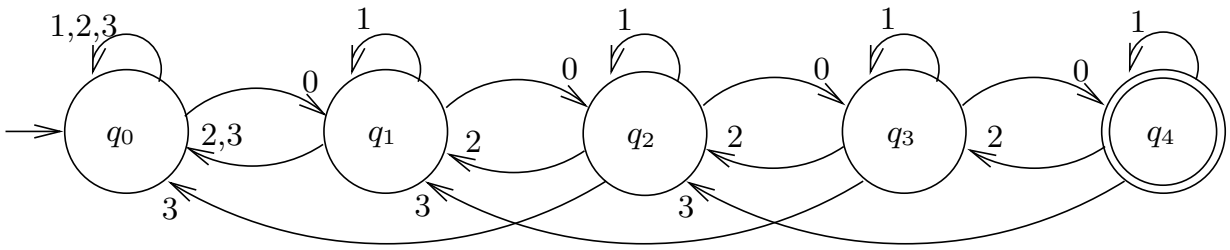

Figure 4.1: An example of a ruler automaton: the DFA $R(5)$.

The relationship between the ruler automata and the ruler sequence is apparent from the simplicity of the transition functions on prefixes of the ruler sequence, and from each ruler automaton accepting exactly one prefix of the ruler sequence.

Proposition 47 *For any integers $i \geq 0, k \geq 1, n \geq i + k$, $\delta_n(q_i, r_k) = q_{i+k}$.*

Proof: We prove this by induction. First observe that the base case is true: for $k = 1$ we get $r_k = 0$, and $\delta_n(q_i, 0) = q_{i+1}$. Now assume that the claim is true for all $k \leq j$, and that we thus have $\delta_n(q_i, r_j) = q_{i+j}$. Let us show that it holds for $k = j + 1$.

Since $r_{j+1} = r_j j r_j$, we know that

$$\begin{aligned}
\delta(q_i, r_{j+1}) &= \delta(q_i, r_j j r_j) \\
&= \delta(\delta(\delta(q_i, r_j), j), r_j) \\
&= \delta(\delta(q_{i+j}, j), r_j) \\
&= \delta(q_{(i+1)}, r_j) \\
&= q_{i+j+1}.
\end{aligned}$$

□

Corollary 48 *The only prefix of the ruler sequence accepted by $R(n)$ is r_{n-1}.*

The relationship between the ruler automata and Pascal's triangle comes from the number of times each state is reached in $R(n)$ by r_{n-1}.

Definition 49 *We define the following three functions that map from a 4-tuple (M, q_a, q_i, u) to a nonnegative integer, where M is an FA, q_a is a state of interest, q_i is a source state, and $u \in \Sigma^*$.*

- *Let* $\text{enter}(M, q_a, q_i, u)$ *be the number of times the state q_a is entered in M when the word u is read from the state q_i. If either of q_a or q_i is not a state in M, then the function evaluates to zero.*
- *Let* $\text{exit}(M, q_a, q_i, u)$ *be the number of times the state q_a is exited in M when the word u is read from the state q_i. If either of q_a or q_i is not a state in M, then the function evaluates to zero.*
- *Let* $\text{count}(M, q_a, q_i, u) = \max(\text{enter}(M, q_a, q_i, u), \text{exit}(M, q_a, q_i, u))$.

So $\text{count}(M, q_a, q_i, u)$ is the number of times state q_a is reached in M by word u read from state q_i. We include starting in state q_i as reaching it once, and any states not defined in M are counted as being reached zero times.

Proposition 50 *For any integers* $a, i \geq 0, k \geq 1, n \geq i + k$, $\text{count}(R(n), q_a, q_i, r_k) = \text{count}(R(n), q_{a-1}, q_i, r_{k-1}) + \text{count}(R(n), q_a, q_i, r_{k-1})$.

Proof: Here δ_n refers to the transition function of $R(n)$. First note that if u and v are words, and x is a symbol, then

$$\text{count}(R(n), q_a, q_i, uxv) = \text{count}(R(n), q_a, q_i, u) + \text{count}(R(n), q_a, \delta_n(q_i, ux), v).$$

Since $r_k = r_{k-1}(k-1)r_{k-1}$, and $\delta_n(q_i, r_{k-1}(k-1)) = q_{i+1}$, we have

$$\begin{aligned}\text{count}(R(n), q_a, q_i, r_k) &= \text{count}(R(n), q_a, q_i, r_{k-1}) + \text{count}(R(n), q_a, \delta_n(q_i, r_{k-1}(k-1)), r_{k-1}) \\ &= \text{count}(R(n), q_a, q_i, r_{k-1}) + \text{count}(R(n), q_a, q_{i+1}, r_{k-1}).\end{aligned}$$

Observe that for $a, i, j \geq 0, k \geq 1, n \geq i + k$, and $n \geq j + k$, we have

$$\text{count}(R(n), q_{i+a}, q_i, r_k) = \text{count}(R(n), q_{j+a}, q_j, r_{k-1}).$$

If $j = i - 1$, then

$$\text{count}(R(n), q_a, q_{i+1}, r_{k-1}) = \text{count}(R(n), q_{a-1}, q_i, r_{k-1}).$$

So we have

$$\text{count}(R(n), q_a, q_i, r_k) = \text{count}(R(n), q_{a-1}, q_i, r_{k-1}) + \text{count}(R(n), q_a, q_i, r_{k-1}).$$

□

So, the number of times the state q_i is reached in $R(n)$ by r_{n-1} is the sum of the number of times the states q_{i-1} and q_i are reached in $R(n-1)$ by r_{n-2}. Hence, counting the number of times each state in $R(n)$ is reached by r_{n-1} generates Pascal's triangle. In particular, the number of times the q_i state is reached in $R(n)$ by r_{n-1} is $\binom{n-1}{i-1}$.

Originally the author thought that the shortest squarefree word accepted by $R(n)$ was r_{n-1}, and that these automata hence provided an exponential lower bound on the length of the shortest squarefree word accepted by a DFA. However, the author later found that these automata accept shorter squarefree words over the symbols $\{0, 1, 2\}$. Shallit [37] found the shortest squarefree word accepted by $R(7)$ to be of length 40:

$$0102101201020121020102101201020121012010.$$

In general the shortest squarefree word accepted by $R(n)$ appears to be the prefix of some longer sequence that Shallit found.

Our next problem was recently asked in a paper [34] on the computational complexity of some universality problems, but had been studied earlier by Restivo [35].

Open Problem 51 *What is the computational complexity of determining, given a finite set of finite words $S \subset \Sigma^*$, whether* $\mathrm{Fact}(S^*) = \Sigma^*$*?*

Deciding universality for a language $L \subseteq \Sigma^*$ is the problem of determining whether $L = \Sigma^*$. We refer to the set of prefixes of L as

$$\mathrm{Pref}(L) = \{x \in \Sigma^* : \exists y \in L \text{ such that } x \text{ is a prefix of } y\}.$$

We use $\mathrm{Suff}(L)$, $\mathrm{Fact}(L)$, and $\mathrm{Subw}(L)$ in the analogous way for suffixes, factors, and subwords, respectively. Rampersad et al. [34] give the computational complexity for solving the universality problem for $\mathrm{Pref}(L)$, $\mathrm{Suff}(L)$, $\mathrm{Fact}(L)$, and $\mathrm{Subw}(L)$ where L is given by a DFA or NFA, and then go on to consider the case where L is a finite set of finite words. The proof of the following proposition is based on one from Rampersad et al. [34].

Proposition 52 *We can test in O(n) time, where $n = \Sigma_{w \in S}|w|$, whether a finite set of finite words S has the property that* $\mathrm{Pref}(S^*) = \Sigma^*$.

Proof: We begin by building a DFA $M = (Q, \Sigma, q_0, F, \delta)$ that accepts all words in S. Since S is a finite set of finite words we can do this by inserting each word, one at a time, into a trie-shaped DFA. For each $i \in \Sigma$, we let $l_i \in Q$ be the state such that $\delta(q_0, i) = l_i$. Then we alter the transition function such that for each $i \in \Sigma$ and each $f \in F$, we get $\delta(f, i) = l_i$. Subsequently we remove all unreachable states and add each remaining state to F. Any undefined transitions at this point are sent to a rejecting state, r. Call this resulting DFA M'. For an example of this construction, see Figure 4.2.

Let $S' \subseteq S$ be the largest set of words obtained by removing words from S that have proper prefixes also in S. Then $L(M') = \mathrm{Pref}(S'^*)$, and if $L(M') = \Sigma^*$ then also $\mathrm{Pref}(S^*) = \Sigma^*$. If $L(M') \neq \Sigma^*$ then there must be a rejecting state and a word w that reaches it. This w cannot be in $\mathrm{Pref}(S^*)$. It follows that $L(M') = \Sigma^*$ if and only if

$\text{Pref}(S^*) = \Sigma^*$. The DFA M' can be built in linear time and the presence of a rejecting state can be checked in linear time, so the proposition is true. □

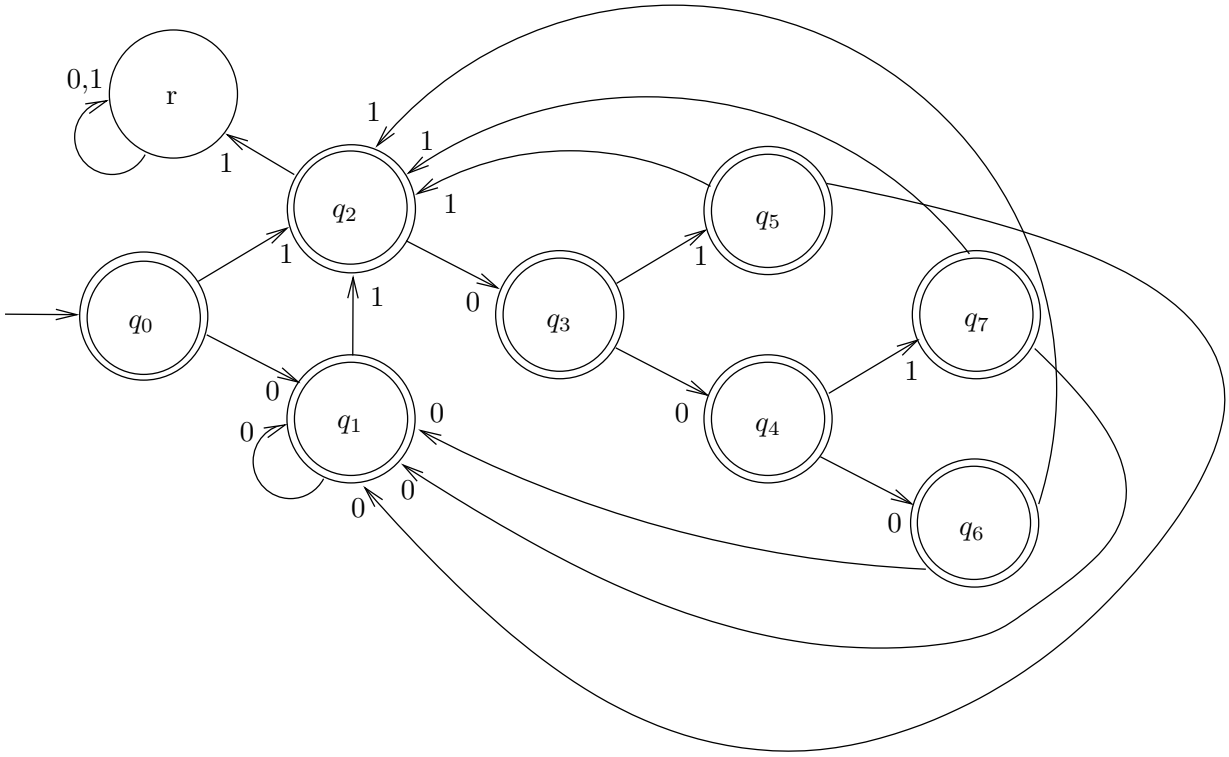

Figure 4.2: An example of DFA M' for $S = \{0, 010, 101, 1001, 1000\}$. Since $\delta(q_2, 1) = r$, we know that the word 11 is not in $\text{Pref}(S^*)$.

While the algorithm in the preceding proof can also be used to check if $\text{Suff}(S^*) = \Sigma^*$ by reversing each word in S before running the algorithm, unfortunately this algorithm cannot easily be adapted to determine if $\text{Fact}(S^*) = \Sigma^*$. This leads to Open Problem 51. When studying this problem, Restivo [35] found a linear time solution for the case where the finite set of finite words $S \subset \Sigma^*$ is restricted to being a code: $\text{Fact}(S^*) = \Sigma^*$ if and only if $\Sigma_{w \in S} |\Sigma|^{-|w|} = 1$. For the general case it is not hard to see that the problem can be solved with polynomial space:

Proposition 53 *Given a finite set of finite words $S \subset \Sigma^*$, the problem of determining whether* $\text{Fact}(S^*) = \Sigma^*$ *is in PSPACE where the input size is $n = \Sigma_{w \in S} |w|$.*

Proof: We begin by building a DFA $M = (Q, \Sigma, q_0, F, \delta)$ that accepts all words in S. This can be done in $O(n)$ time with $|Q| \in O(n)$. Then we convert M into an ϵ-NFA M' that accepts S^* by adding ϵ-transitions from each final state to the initial state. Rampersad et al. [34] show that it is PSPACE-complete in general to check if $\text{Fact}(L(M')) = \Sigma^*$ for an NFA M'. □

While it may be the case the the particular structure of the NFA M' in the immediately preceding proof allows for a faster method to check if $\text{Fact}(L(M')) = \Sigma^*$, it is not clear whether this is so. Another approach that the author has attempted is to rewrite $\text{Fact}(S^*)$ as a concatenation of the languages $S^*, \text{Pref}(S), \text{Suff}(S), \text{Pref}(S^*)$, and $\text{Suff}(S^*)$ with the hope of finding a way to combine solutions to the universality problem for these different languages.

Proposition 54 *For a language L, the following four languages are equivalent to* $\text{Fact}(L^*)$*:*

1. $\text{Suff}(L)\text{Pref}(L^*)$
2. $\text{Suff}(L^*)\text{Pref}(L)$
3. $\text{Suff}(L)L^*\text{Pref}(L)$
4. $\text{Suff}(L^*)\text{Pref}(L^*)$

Proof: Since the only difference between $\text{Pref}(L^*)$ and $\text{Fact}(L^*)$ is that a word in $\text{Fact}(L^*)$ can begin with a proper suffix of a word in L, we get $\text{Fact}(L^*) = \text{Suff}(L)\text{Pref}(L^*)$. By symmetry we have $\text{Fact}(L^*) = \text{Suff}(L^*)\text{Pref}(L)$. The other two equivalent ways of expressing $\text{Fact}(L^*)$ follow from the observation that $\text{Pref}(L^*) = L^*\text{Pref}(L)$ and $\text{Suff}(L^*) = \text{Suff}(L)L^*$.

□

Unfortunately, the author has been unable to find a way to use the different ways of expressing $\text{Fact}(S^*)$ to find a better upper bound on the computational complexity than the PSPACE bound provided in Proposition 53.

Another approach to determining whether $\text{Fact}(S^*) = \Sigma^*$ is to systematically check for the shortest word not in $\text{Fact}(S^*)$. It is not hard to build an NFA that accepts $\text{Fact}(S^*)$ in $O(n)$ time where $n = \Sigma_{w \in S}|w|$, so the efficiency of this brute force method depends on the

length of the shortest string not in Fact(S^*). This leads to the following shortest string problem.

Open Problem 55 *Given a finite set of finite words S, what are good bounds on the length of the shortest string not in* Fact(S^*)*?*

Both Pribavkina [32] and Rampersad et al. [34] have independently established the following lower bound with respect to the length of the longest word in S.

Proposition 56 *For each integer $n \geq 1$ there exists a set of finite words of length $\leq n$, such that the shortest word not in* Fact(S^*) *is of length $n^2 + n - 1$.*

Restivo [35] conjectured an upper bound of $2n^2$, but in a paper by Fici et al. [19] this conjecture is shown to be false. Fici et al. [19] take a closer look at the structure of shortest words not in Fact(S^*) and construct some examples that are quadratic in the length of the longest string in S.

References

[1] T. Anderson, J. Loftus, N. Rampersad, N. Santean, and J. Shallit. Detecting palindromes, patterns and borders in regular languages. *Inf. Comput.*, 207(11):1096–1118, 2009. 42

[2] T. Ang and J. Brzozowski. Languages convex with respect to binary relations, and their closure properties. *Acta Cybernetica*, 19(2):445–464, 2009. 4

[3] T. Ang, G. Pighizzini, N. Rampersad, and J. Shallit. Automata and reduced words in the free group, preprint. http://arxiv.org/abs/0910.4555, 2009. iii, 25, 26, 30, 35, 39, 40

[4] T. Ang and J. Shallit. Length of the shortest word in the intersection of regular languages, preprint. http://arxiv.org/abs/0910.1528, 2009. iii, 17

[5] M. Benois. Parties rationnelles du groupe libre. *C. R. Acad. Sci. Paris Sér. A-B*, 269:A1188–A1190, 1969. 9, 24

[6] M. Benois and J. Sakarovitch. On the complexity of some extended word problems defined by cancellation rules. *Inf. Process. Lett.*, 23(6):281–287, 1986. 9, 22

[7] J. Berstel. *Transductions and Context-Free Languages.* Teubner, 1979. 9

[8] J. Berstel. Some recent results on squarefree words. In *STACS '84: Proceedings of the Symposium of Theoretical Aspects of Computer Science*, pages 14–25, London, UK, 1984. Springer-Verlag. 42, 43

[9] J. Berstel. Axel Thue's work on repetitions in words. In P. Leroux and C. Reutenauer, editors, *Séries Formelles et Combinatoire Algébrique*, Publications du LACIM 11, pages 65–80, Université du Québec, Montréal, 1992. 42

[10] J.-C. Birget. Partial orders on words, minimal elements of regular languages, and state complexity. *Theor. Comput. Sci.*, 119(2):267–291, 1993. 6

[11] L. Boasson, B. Courcelle, and M. Nivat. The rational index: a complexity measure for languages. *SIAM Journal on Computing*, 10(2):284–296, 1981. 5, 9, 30, 39

[12] R. V. Book. Confluent and other types of Thue systems. *J. Assoc. Comput. Mach.*, 29(1):171–182, 1982. 9, 26

[13] R. V. Book. Homogeneous Thue systems and the Church-Rosser property. *Discrete Mathematics*, 48(2-3):137–145, 1984. 9, 20

[14] R. V. Book and F. Otto. Cancellation rules and extended word problems. *Inf. Process. Lett.*, 20(1):5–11, 1985. 9, 20, 22

[15] J. A. Brzozowski. Quotient complexity of regular languages. In J. Dassow, G. Pighizzini, and B. Truthe, editors, *Descriptional Complexity of Formal Systems*, pages 25–42, Magdeburg, Germany, 2009. 5

[16] C. Câmpeanu, K. Culik II, K. Salomaa, and S. Yu. State complexity of basic operations on finite languages. In *WIA '99: Revised Papers from the 4th International Workshop on Automata Implementation*, pages 60–70, London, UK, 2001. Springer-Verlag. 5

[17] K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: new results and open problems. *J. Autom. Lang. Comb.*, 10(4):407–437, 2005. 5

[18] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W.P. Thurston. *Word Processing in Groups.* Jones and Bartlett Publishers, 1992. 9

[19] G. Fici, E. V. Pribavkina, and J. Sakarovitch. On the minimal uncompletable word problem, preprint. http://arxiv.org/abs/1002.1928, 2010. 49

[20] S. Ginsburg and E. H. Spanier. Quotients of context-free languages. *J. Assoc. Comput. Mach.*, 10:487–492, 1963. 25

[21] I. Glaister and J. Shallit. A lower bound technique for the size of nondeterministic finite automata. *Inf. Process. Lett.*, 59(2):75–77, 1996. 5, 6

[22] H. Gruber and M. Holzer. Finding lower bounds for nondeterministic state complexity is hard (extended abstract). In O. H. Ibarra and Z. Dang, editors, *Proceedings of the 10th International Conference on Developments in Language Theory*, Lect. Notes Comput. Sci., 4036:363–374. Springer, 2006. 5, 6

[23] J. Gruska. Descriptional complexity (of languages) - a short survey. In A. Mazurkiewicz, editor, *Proc. 5th Symposium, Mathematical Foundations of Computer Science*, Lect. Notes Comput. Sci., 45:65–80, 1976. 1

[24] Y. Han, K. Salomaa, and S. Yu. State complexity of combined operations for prefix-free regular languages. In *LATA '09: Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, pages 398–409, Berlin, Heidelberg, 2009. Springer-Verlag. 5

[25] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979. 1, 3, 4, 5, 8, 11, 22, 27

[26] G. F. Italiano. Amortized efficiency of a path retrieval data structure. *Theor. Comput. Sci.*, 48:273–281, 1986. 24

[27] A. N. Maslov. Estimates of the number of states of finite automata. *Dokl. Akad. Navk. SSSR.*, 194:1266–1268, 1970. In Russian. English translation in *Soviet Math. Dokl.* 11:1373–1375, 1970. 5

[28] 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(10):1211–1214, 1971. 5, 28

[29] L. Pierre and J. Farinone. Rational index of context-free languages in exp $\theta(\sqrt[p]{n})$ and $n^{\Theta((\ln n)^{\frac{1}{p}})}$. *Theor. Comput. Sci.*, 57(2-3):185–204, 1988. 9

[30] G. Pighizzini. Personal communication, 2010. 27, 28

[31] J. A. Poutré and J. van Leeuwen. Maintenance of transitive closures and transitive reductions of graphs. In H. Gottler and H.J. Schneider, editors, *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science.*, Lect. Notes Comput. Sci., 314:106–120. Springer-Verlag, 1998. 24

[32] E. V. Pribavkina. Slowly synchronizing automata with zero and incomplete sets, preprint. http://arxiv.org/abs/0907.4576, 2009. 49

[33] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959. 5, 8

[34] N. Rampersad, J. Shallit, and Z. Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages, preprint. http://arxiv.org/abs/0907.0159, 2009. 46, 48, 49

[35] A. Restivo. Some remarks on complete subsets of a free monoid. In *Quaderni de La Ricerca Scientifica*, CNR Roma, 109:19–25, 1981. 46, 47, 49

[36] J. Shallit. State complexity and Jacobsthal's function. In *CIAA '00: Revised Papers from the 5th International Conference on Implementation and Application of Automata*, pages 272–278, London, UK, 2001. Springer-Verlag. 5, 9, 12

[37] J. Shallit. Personal communication, 2009. 14, 27, 45

[38] S. Yu, Q. Zhuang, and K. Salomaa. The state complexities of some basic operations on regular languages. *Theor. Comput. Sci.*, 125(2):315–328, 1994. 4