

LIMITED EXISTENTIAL AND UNIVERSAL WIDTH
ALTERNATING FINITE AUTOMATA

by

MOHAMMAD ZAKZOK

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada

April 2024

Copyright © Mohammad Zakzok, 2024

Abstract

There have been many studies on several models of limited nondeterministic finite automata (NFAs), including studies on tree width, branching, guessing, and ambiguity of NFAs. However, the study of limited alternating finite automata (AFAs), a generalization of NFAs that allows for alternation between existential and universal states, is fairly new in the literature. Drawing inspiration from limited NFAs, recent studies have introduced maximal and optimal existential and universal widths of AFAs, designed to quantify the amount of existential and universal power used by an AFA.

In this thesis, I explore the relation between limited optimal existential and universal width AFAs, limited maximal existential and universal width AFAs, NFAs, universal finite automata (UFAs), and deterministic finite automata (DFAs). More specifically, the study focuses on the state complexity of those models and the way they relate to each other. I show improved upper bounds for converting limited existential and/or universal width AFAs into NFAs, UFAs, and DFAs, largely using improved variations of subset construction. I also establish lower bounds for the size of NFAs, UFAs, and DFAs simulating limited existential and/or universal width AFAs. Although the upper and lower bounds provided are not the same, they are reasonably close.

We see that although existential and universal width are not as symmetric as initially perceived, restricting both existential and universal maximal widths does enable us to circumvent some of those symmetry issues. Lastly, I explore relations between the measure of existential width of NFAs and the classical measures of tree width, branching, and guessing of NFAs.

Acknowledgments

I would like to thank Professor Kai Salomaa for his invaluable guidance, help, and patience in the research of this thesis. We were able to accomplish our goals and a bit more despite facing many difficulties. Originally, I had not anticipated that we would be able to publish the results within such a short period of time and graduate, but somehow he made it possible. I've already learned a lot from Prof. Salomaa, but I believe there is still more to come.

I would also like to thank the defence committee, Professor Selim Akl and Professor Sean Kauffman, for taking the time to go through this thesis and providing valuable feedback, as well as Professor Ahmed Hassan for chairing the committee. A special thanks to Debby Robertson for her support and for arranging the defence on such short notice. I would also like to express gratitude to Queen's University and the School of Computing for providing me with this opportunity.

Lastly, I would like to express my gratitude to Professor Christos Kapoutsis for inspiring me to study finite automata, as well as to my friends, especially Sana. A last but not least special thanks to my parents and family, especially Momina, for their continued support.

Contents

| | |
|---|------------|
| Abstract | i |
| Acknowledgments | iii |
| Contents | iv |
| List of Figures | vi |
| Chapter 1: Introduction | 1 |
| 1.1 Motivation and Literature Review | 1 |
| 1.1.1 What is State Complexity? | 1 |
| 1.1.2 Why State Complexity? | 3 |
| 1.1.3 Literature Review: Alternation | 4 |
| 1.1.4 Literature Review: Limiting NFAs | 6 |
| 1.1.5 Literature Review: Limiting AFAs | 8 |
| 1.2 Research Outline | 9 |
| Chapter 2: Preliminaries | 13 |
| 2.1 Deterministic and Nondeterministic Finite Automata | 13 |
| 2.2 Measures of Nondeterminism | 15 |
| 2.3 Alternating Finite Automata | 16 |
| 2.4 Existential and Universal Width of AFAs | 19 |
| 2.5 Fooling Sets | 23 |
| Chapter 3: Limited Universal Width Alternating Finite Automata | 26 |
| 3.1 Finite Universal Width AFAs and NFAs | 26 |
| 3.1.1 Upper Bounds | 29 |
| 3.1.2 Lower Bounds | 32 |
| 3.2 Finite Universal Width UFAs and DFAs | 37 |
| 3.2.1 Upper Bounds | 37 |
| 3.2.2 Lower Bounds | 40 |

| | | |
|---------------------|--|-----------|
| Chapter 4: | Limited Existential Width Alternating Finite Automata | 42 |
| 4.1 | Finite Existential Width AFAs and NFAs | 42 |
| 4.1.1 | Differences Between Existential and Universal Width | 43 |
| 4.1.2 | Upper Bounds | 45 |
| 4.1.3 | Lower Bounds | 48 |
| 4.2 | Finite Existential Width NFAs and DFAs | 53 |
| 4.2.1 | Upper Bounds | 53 |
| 4.2.2 | Lower Bounds | 54 |
| 4.3 | Properties of NFAs with Limited Existential Width | 56 |
| Chapter 5: | Limited Existential and Universal Width Alternating Finite Automata | 61 |
| 5.1 | Limited Combined Width AFAs | 61 |
| 5.2 | Limited Universal Width AFAs | 64 |
| 5.3 | Separately Limited Widths | 65 |
| Chapter 6: | Summary and Conclusion | 66 |
| 6.1 | Conclusions | 67 |
| 6.2 | Future Work | 67 |
| Bibliography | | 69 |

List of Figures

| | | |
|-----|---|----|
| 2.1 | AFA A recognizing the language $\{11\}$ | 17 |
| 2.2 | AFA A recognizing the language $\text{Divisible}(14, 15)$ | 20 |
| 2.3 | Pruned computation tree of an AFA | 22 |
| 2.4 | AFA A recognizing ab^* | 23 |
| 3.1 | An AFA A | 28 |
| 3.2 | An NFA B recognizing $L(A)$ | 28 |
| 3.3 | Visualising liveness for $h = 5$ and a string of length 3. | 33 |

Chapter 1

Introduction

1.1 Motivation and Literature Review

In theoretical computer science, the difference between classical computability theory and complexity theory is commonly understood. Computability theory asks questions around which problems can be solved by a computational model, classically Turing machines. Formally, such a problem is said to be *decidable* by a Turing machine. On the other hand, complexity theory asks questions about which problems can be solved within certain restrictions, most commonly space and/or time, by a Turing Machine. The utility of those studies is fairly obvious as the Turing machine model is the most commonly used to represent modern computational capabilities.

1.1.1 What is State Complexity?

The same distinction between computability and complexity can be made on the level of finite automata, although the distinction between the two is far less obvious. Informally, a finite automaton essentially models a computer that has only a constant amount of writable memory. It is fairly straightforward to understand what it means

for a problem to be decidable by a finite automaton, a problem is decidable by a finite automaton if it can be solved by a finite automaton. However, it is more complicated to understand the study of complexity theory on the level of finite automata as any finite automaton does not use space and runs in linear time. Instead, the complexity of finite automata often studies the state complexity of the machine [5, 14, 19, 21]. That is to say, problems are gathered in classes quantifying the number of states it takes to solve a problem. There are also other measures that are less studied such as transition function [10, 46] and alphabet size, but those measures are not of interest in this study and are generally rarely studied.

Still, it is not exactly clear what studying the size of automata exactly entails. After all, given a problem, or a *language*, L , any automaton that solves, or *recognizes*, L has a constant number of states. For this reason, we study language families. For example, consider a language family $\{L_1, L_2, \dots, L_n, \dots\}$, where L_n is simply the regular expression a^n for $n \in \mathbb{N}$. Any *deterministic finite automaton* (DFA) recognizing L_n clearly requires n states to count the number of a 's in the input. The proof for this statement is fairly simple, but we do not go over it in the introduction. Formally, we say that any DFA recognizing L_n requires n states.

Nevertheless, it may not seem obvious to the reader why we study the state complexity of automata at all, as finite automata rarely seem to be useful in real world scenarios. Of course, finite automata are useful in conceptualizing real world problems and have direct applications in automata based programming [38], design of object oriented software [11], and software specification [47]. However, there are other reasons for studying the state complexity of finite automata.

1.1.2 Why State Complexity?

State complexity is often used to simplify or distill certain aspects of classical complexity theory down to their very core. Let us, for example, consider the biggest and most important problem in complexity theory, $P \stackrel{?}{=} NP$, introduced in the early 70s [9, 37]. I am sure the reader is aware of the infamous $P \stackrel{?}{=} NP$ problem and its ramifications, but I give a brief reminder of the problem for the purpose of relating our problems to the reader. In plain terms, P is the class problems solved by a Turing machine that runs in at most polynomial time while NP is the class of problems whose solutions can be verified in polynomial time. Alternatively and perhaps less intuitively, NP could be defined the class problems which can solved by a *nondeterministic* Turing machine. In loose terms, a nondeterministic Turing machine is a Turing machine that has the ability to duplicate itself infinitely, *accepting* the input if any duplicate of the machine accepts it.

Distilling $P \stackrel{?}{=} NP$

As the problem has proven to be stubbornly difficult, theoretical computer scientists have tried distilling certain aspects of the problem in hope of making a partial breakthrough that may lead to better intuitive understanding and thus bigger results down the line. For example, consider the problem $L \stackrel{?}{=} NL$ [40, 44], which compares a deterministic and non-deterministic Turing machine with at most logarithmic space. The purpose of studying $L \stackrel{?}{=} NL$ is to simplify the time and space complexity aspects of $P \stackrel{?}{=} NP$ to allow further focus on comparisons between determinism and nondeterminism, which seem to be at the crux of $P \stackrel{?}{=} NP$. However, the problem $L \stackrel{?}{=} NL$ also proves to be stubbornly difficult and remains open.

A further simplification exists at the automata theory level, introduced by Sakoda and Siper in 1978 [43]. The problem asks the question of whether deterministic and nondeterministic polynomial sized two-way automata solve the same set of problems. In simpler terms, the problem explores the $P \stackrel{?}{=} NP$ for Turing machines that do not have any memory, which incidentally makes them two-way automata. This further distills the problem to purely pit determinism and nondeterminism against each other. Unfortunately, even this problem remains unsolved.

1.1.3 Literature Review: Alternation

Alternation in Complexity Theory

As this thesis largely studies properties of *alternating finite automata* (AFAs), it is only natural to understand the origin of alternation. Alternation was introduced in 1976 independently by Chandra and Stockmeyer [7] and Kozen [33], and later combined in 1981 [6]. Alternation gives machines the ability to switch between *existential* (nondeterministic) and *universal* powers. In intuitive terms, universal powers are inverse to existential powers. Following our previous example of duplicating machines, a machine with universal powers is able duplicate itself infinitely into many machines, but accepts the input if **all** branches accept. A machine with both existential and universal powers is able to alternate between those two modes, giving it far more power.

Alternation is one of two ways used to model one of most important ideas in complexity theory, the *polynomial time hierarchy* (*PH*) [40, 44], a generalization of the class *NP*. As the other model uses the idea of oracles and is more intuitive to the reader, I first describe *PH* using oracles. The hierarchy is defined inductively,

with the first level being the class NP . The second level, NP^{NP} is defined as the set of problems solvable by a polynomial time nondeterministic Turing machine with access to a black box oracle that can solve any NP -hard problem. The third level is the set of problems solvable by a polynomial time nondeterministic Turing machine with access to an oracle solve any NP^{NP} -hard problems... and so it goes on infinitely.

We can also define PH using alternation [2]. The first level retains the same definition, the class NP . The second level, NP^{NP} , is defined as the set of problems solvable by a polynomial time nondeterministic Turing machine with one alternation. The third level adds the ability to alternate two times, and so on. It is not obvious why those two definitions of PH are equivalent, but we do not need to understand the reasoning for the purposes of this study. It is not known whether PH is strict. That is to say, it is not known whether any two levels in PH are equivalent. However, it is has been shown that if any two levels were equivalent, then all levels above them would also collapse down to the lower level of the two [2].

Alternation in Automata Theory

Alternating finite automata (AFAs) were introduced in the same work that introduced alternating Turing machines in 1981 [6]. However, the model originally discussed was a slightly different model of automata allowing for logical gates representation of states, which differs from our model as our model only allows for **AND** and **OR** gates. They also showed that the upper bound for simulating an AFA of size n with an NFA is 2^n . The same result does hold for the model of AFAs used in this thesis. Moreover, in symmetric fashion, the upper bound for simulating an AFA of size n with an automaton that has exclusively universal powers is 2^n . In our research, we

improve both of those two bounds for more specific scenarios.

Parallels to PH do exist on simplified automata level, although defining them is not as straightforward. As automata are not able to write, defining how oracle finite automata would work requires far more work. However, defining a polynomial size hierarchy using alternation is far easier, as adding alternation powers to automata is very 'plug and play'. Unlike PH , the polynomial size alternating two-way automata hierarchy, $2A$ has been shown to be strict [12], showcasing that we could learn more about alternation by simplifying it to the level of automata. Furthermore, a polynomial size alternating one-way automata hierarchy, $1A$, has also been shown to be strict and even has an equivalent definition that uses oracles [1, 27].

1.1.4 Literature Review: Limiting NFAs

Nondeterministic finite automata (NFAs) are one of the most popular classically studied computation model in automata theory and state complexity. NFAs with limited nondeterminism were first introduced in 1971 [32] but have since been studied meticulously in numerous studies, as showcased in two survey studies [14, 19]. We go over some of those studies and results in a very brief manner. In this study, we use the terms nondeterministic and existential interchangeably. The term nondeterministic was originally used to mean the inverse of deterministic [41] before the addition of alternation [6]. However, the introduction of alternation makes both universal and existential to mean the inverse of deterministic. In general, studies centered around NFAs and DFAs will only use the term nondeterministic while studies centered around alternation will favour the terms existential and universal.

Ambiguity

Introduced in 1971 by Book et al. [4] and more systematically studied in Ravikumar and Ibarra [42], *ambiguity* is the first measure used to limit the powers of NFAs. An ambiguous NFA has multiple accepting branches while an unambiguous machine has exactly one accepting branch for each accepting computation. The degree of ambiguity is used to describe the total number of accepting computations on any input for an NFA. It has been shown that verifying the equivalence of finitely ambiguous NFAs can be done in polynomial time [45, 48].

Moreover, it has been shown that classifying the level of ambiguity of an NFA into finitely ambiguous, strictly polynomially ambiguous, or strictly exponentially ambiguous can be done in polynomial time [20, 48]. However, finding the exact quantity of ambiguity of NFA has been shown to be *P-SPACE* complete [24]. Lastly, it has been shown that the lower bound for simulating any unambiguous finite automaton of size n using a DFA is $2^n - 1$ [34, 35]. Still, ambiguity does not limit the amount of nondeterminism used in an NFA, but rather the number of accepting computations. For this reason, it is not as relevant to this research as methods of limiting nondeterminism.

Limiting Nondeterminism

There are multiple ways of studying limited nondeterminism in NFAs. The first and most obvious studies NFAs with limited tree size. Consider an NFA A and an input w , the amount of nondeterminism of A on w is the tree width of the computation tree of A on w , denoted as $tw_A(w)$ [14, 15, 23, 39]. Goldstine et al. [14] also formalize other aspects such as the *guessing* and *branching* of an NFA. The guessing of A

is used to quantify the total number of times A made nondeterministic 'guesses' in one specific computation branch, while the branching measures the total number of resulting branches in the computation. The guessing and branching of A on w can both be defined optimally or maximally, depending on whether we consider the most optimal or least optimal computation of A on w . Hromkovic et al. [23] would go on to prove that the tree width of an NFA is either bounded by a constant, or between linear and polynomial in m , or otherwise in $2^{\Theta(m)}$, where m is the size of the input [19].

1.1.5 Literature Review: Limiting AFAs

As AFAs have far more complicated definitions of acceptance, defining limited width measures also become more complicated. For this reason, for an AFA A , we consider width measures on *witness partial computation trees*. Essentially, for A to accept an input w , the computation tree of A on w must contain a partial tree that can sufficiently witness the acceptance of w by A . In our study, we use the terminology *pruned computation trees*. As defining this process is fairly dense, we do not go into it here, leaving it for chapter 2.

Drawing large inspiration from measures of nondeterminism on NFAs, recent studies have considered similar limitations on AFAs [17, 31]. In those studies, two main ideas are introduced, the optimal existential and universal widths of AFAs, designed with the idea of quantifying the number of existential and universal guesses taken by an AFA. The optimal existential width roughly measures the number of existential guesses made in an accepting witness partial computation tree, including existential

branches that are not followed in the partial tree. Meanwhile, the optimal universal width roughly quantifies the total number of branches required in an accepting witness partial computation tree. Furthermore, the studies also introduce maximal existential and universal width measures. The maximal existential width roughly measures the number of existential guesses made in any witness partial computation tree, regardless of whether it is accepting or not. Similarly, the maximal universal width roughly quantifies the total number of branches required in any witness partial computation tree.

1.2 Research Outline

Our main goal in this study was to perform a state complexity comparison between AFAs with limited existential and/or universal width and other classical models such as NFAs, universal finite automata (UFAs), and DFAs. We also seek to compare those new limitations to older limitations such as tree width, branching, and guessing of NFAs. A large portion of the study's results is published in the *Theoretical Computer Science* [49] journal. In that paper, we publish most of our results comparing the state complexity NFAs and UFAs to AFAs with limited existential and/or universal width.

In chapter 3, we explore relations between AFAs with limited optimal and maximal universal width to NFAs, DFAs, and unary DFAs. First, we find improved upper and lower bounds for converting limited universal width AFAs to NFAs. For the upper bound, we create an NFA that is able to simulate limited universal width AFAs by using an improved subset construction that relies on the new limitations. We then show that there exists an AFA A with limited universal width u such that any NFA

recognizing the language of A requires at least 2^u states. Although the upper and lower bounds we provide are not exactly the same, they are reasonably close. We are able to obtain results as corollaries for DFAs by using subset construction on the resulting NFAs. We further improve those results for unary DFAs using ideas from [8]. Lastly, similar ideas are used to prove upper and lower bounds for comparing limited universal width UFAs to DFAs.

Initially, we had expected that universal and existential width would be largely symmetrical notions. However, universal width proved to be the far simpler of the two, which is why we discuss it first. The universal width counts the total number leaves in a witness partial computation tree, while the existential width counts the total number of existential guesses not taken in a witness partial computation tree. This creates difficulties when we try to find symmetric proofs and constructions, as the existential width does not accurately represent the total number of existential branches on the way to accepting the input.

In chapter 4, we explore relations between AFAs with limited optimal and maximal existential width to UFAs and DFAs. Unfortunately, due to difficulties mentioned above, we are not able to provide improved bounds for simulating limited optimal existential width AFAs with UFAs. Nonetheless, we are able to provide improved bounds for simulating limited maximal existential width AFAs with UFAs. We also do provide a lower bound, showing that there exists an AFA A with maximal existential width e such that any UFA recognizing the language of A has at least $2^{e/2} - 1$ states. We also present relations between the tree width, branching, and existential width of NFAs. Lastly, we show that NFAs with limited optimal existential width can be converted into NFAs with limited maximal existential width with a small trade-off in

size.

In Chapter 5, we compare AFAs with both limited optimal existential and universal width to AFAs with limited maximal existential and universal width and show that any AFA with limited optimal widths can be simulated by an AFA with limited maximal widths with some blow up in the state size. Lastly, we are able to obtain results for DFAs by using subset construction on previously constructed machines.

Part of our results have already been published in M. Zakzok, K. Salomaa [49]. The overall contributions of this thesis include:

- Developing the U-fooling set technique, used to provide lower bounds for the size of UFAs.
- In Chapter 3, we focus on comparing limited universal width AFAs and UFAs to other models. Specifically, we provide the following bounds for the size blowup in the following conversions:
 - (i) Upper and lower bounds for converting limited optimal and maximal universal width AFAs to NFAs and DFAs.
 - (ii) Upper and lower bounds for converting limited optimal and maximal universal width UFAs to DFAs.
 - (iii) Improved upper bounds for converting limited optimal universal width unary UFAs to DFAs.
- In Chapter 4, we shift our focus to comparing limited existential width AFAs and NFAs to other models. We provide the following bounds for the size blowup in the following conversions:

-
- (i) Upper and lower bounds for converting limited maximal existential width AFAs to NFAs and DFAs.
 - (ii) Upper and lower bounds for converting limited optimal and maximal existential width NFAs to DFAs.
 - (iii) Upper bounds for converting limited optimal existential width NFAs to limited maximal existential width NFAs.
- We also provide direct comparisons between the existential width of NFAs to the classical measures of tree width, branching, and guessing.
 - In Chapter 5, we show an upper bound for the size blowup of converting a limited combined width AFA to a limited maximal existential and universal width AFA.
 - Moreover, we show an upper bound for the size blowup of converting limited optimal universal width AFAs to maximal universal width AFAs.
 - Lastly, we show an upper bound for the size blowup of converting limited optimal existential and universal width AFAs to DFAs.

Chapter 2

Preliminaries

To properly define our problems and their solutions, we must first mathematically define alternating finite automata, shorthand to AFAs. Although I assume the reader is somewhat familiar with finite automata and basic graph theory, we start by defining the more basic definitions. Σ is defined as a finite alphabet made up of a set of symbols $b \in \Sigma$. Σ^* is set of all strings that can be constructed out of symbols in Σ .

2.1 Deterministic and Nondeterministic Finite Automata

To better understand alternating finite automata, we must first define nondeterministic finite automata. A *nondeterministic finite automaton* (NFA) is a 5-tuple, $A = (Q, \Sigma, \delta, q_0, F)$ where Q is the finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. A *deterministic finite automaton* (DFA) is a restricted NFA such that for all $q \in Q$ and $b \in \Sigma$, $|\delta(q, a)| \leq 1$. In simpler terms, a DFA is an NFA where all transitions are of size at most 1. In the next section, we introduce the concept of unary DFAs, a further restricted version.

For the sake of simplicity, we allow the overloading of the transition function δ

over strings. For all $q \in Q$, $b \in \Sigma$, $w \in \Sigma^*$, the overloaded transition is defined inductively as

$$\delta(q, bw) = \bigcup_{q' \in \delta(q, b)} \delta(q', w).$$

We denote the language of A as $L(A)$. A string $w \in \Sigma^*$ is in $L(A)$ if $\delta(q_0, w) \cap F \neq \emptyset$. In other words, a string w is in $L(A)$ if A , starting from the start state q_0 and reading w , reaches any final state.

We also define $L(A)$ in an alternative method using computation trees. While this method is not as mathematically succinct, it helps us visualise the computation of NFAs. The *computation tree* of A on w starting from state q , denoted as $T_{A,q,w}$, is a tree structure where the nodes are labeled by state-symbol pair or the fail symbol \perp . For an empty string w , $T_{A,q,w}$ is simply a singular node labeled (q, ϵ) . Otherwise, for all $q \in Q$ and $b \in \Sigma$, if $\delta(q, b)$ is not empty, then $T_{A,q,bw'}$ has the root node (q, b) , whose children are all trees $T_{A,q',w'}$ where $q' \in \delta(q, b)$. Lastly, if $\delta(q, b) = \emptyset$, then $T_{A,q,bw'}$ is a singular node labeled (q, b) with one child labeled \perp .

A computation tree $T_{A,q,w}$ is said to be accepting if any of its leaves is labeled by pair a (q_f, ϵ) , where $q_f \in F$. The computation tree of an NFA A on input w is $T_{A,q_0,w}$. Lastly, A accepts input w if $T_{A,q_0,w}$ is accepting. This definition is simply a direct visualisation of the previous definition of acceptance. Notice that finding a singular series of consecutive nodes from the root of $T_{A,q_0,w}$ down to an accepting leaf (q_f, ϵ) proves that A accepts w . This singular branch is called a *witness* of A 's acceptance of w .

The *tree width* of A on w is the total number of leaves in the tree $T_{A,q_0,w}$. As mentioned previously there are numerous studies [18, 22, 29, 36] on the tree width of NFAs or similar properties. This is particularly important as my thesis is focused

on an expanded notion of width to alternating finite automata. We later define tree width more formally.

In certain scenarios, we are able to improve some of our results if we further restrict our model. In this thesis, one such case arises when proving bounds for DFAs, where the bounds could be improved when considering a restricted unary DFA. A DFA A is unary if the alphabet of A has only one symbol. Unary DFAs have some unique desirable properties, shown in [8], that make them particularly interesting. A unary DFA A is always comprised of two sections, the cycle and the tail:

- The cycle is the portion of A that loops between the same states of A . As the size of the input grows, A must return to a previously visited state, from which the cycle starts. In cases where A does not have a cycle, $L(A)$ can only include constant length strings.
- The tail is portion of A that leads up to the state that starts the cycle.

2.2 Measures of Nondeterminism

As we are interested in comparing our new measures of existentiality to already existing measures of existentiality, we give clear definitions for the already existing measures of *branching*, *guessing*, and *tree width* [14, 19] of NFAs. Note that tree width is also defined as 'leaf size' [23]. First, we define the simplest notion, tree width.

Definition 2.2.1. *The tree width of an NFA A on a string w , denoted as $tw_A(w)$ is number of leaves in the computation tree of A on w . The tree width of A , denoted as $tw(A)$ is the supremum of all $tw_A(w)$, for all w in Σ^* , where Σ is the alphabet of A .*

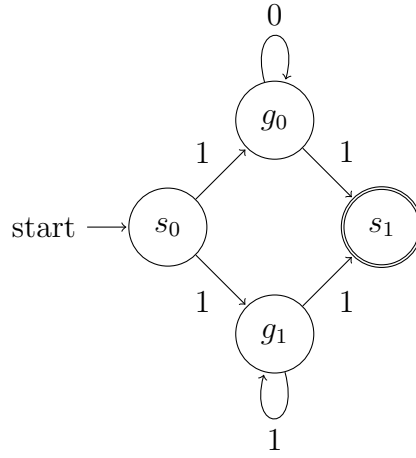
To define the guessing of an NFA A , we must first define the branching of A . The branching is a measure designed to quantify the amount of nondeterminism used by A on a specific branch. The guessing is meant to quantify the branching of A in bits of information.

Definition 2.2.2. *Consider an NFA $A = (Q, \Sigma, \delta, q_0, F)$. The branching of a transition of $\delta(q, b)$ is its size $|\delta(q, b)|$. Consider the computation of A on a string $w = b_0b_1\dots b_m$. The branching of a singular computation $C = \{(q_0, b_0), (q_1, b_1), \dots, (q_m, b_m)\}$ of A on w , such that q_0 is the start state and $q_{i+1} \in \delta(q_i, b_i)$, is the multiplication of the branchings of all $\delta(q_i, b_i)$, denoted by $\beta_{A,w}(C)$. The branching of A on a string w , denoted as $\beta_A(w)$, is the supremum of all $\beta_{A,w}(C)$. Finally, the branching of A , denoted as $\beta(A)$, is the supremum of all $\beta_A(w)$, for all w in Σ^* .*

Definition 2.2.3. *The guessing of A , denoted by $\gamma(A)$, is simply $\log_2(\beta(A))$ and represents the branching of A in bits of information.*

2.3 Alternating Finite Automata

We are now able to expand our definition to alternating finite automata. We use the model of alternating finite automata that does not use logical gates, but has only existential and universal states, introduced in [6] and used by [12, 13, 25, 30]. Our definitions are directly influenced by [17], as my thesis expands on their results. An *alternating finite automaton* (AFA) is a 5-tuple, $A = (Q, \Sigma, \delta, q_0, F)$ where $Q = Q_e \cup Q_u$ is the finite set of states, Q_e is the set of existential states, Q_u is the set of universal states, ($Q_e \cap Q_u = \emptyset$), Σ is the input alphabet, $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states.

Figure 2.1: AFA A recognizing the language $\{11\}$.

We denote the language of A as $L(A)$ and the *size* of A refers to number of states in Q , often denoted by $|Q|$.

I introduce the following notation for simplicity and consistency. If A reads symbol b while in state $g \in Q_e$, we say that A *guesses* states $\delta(g, b)$. On the other hand, if A reads symbol b while in state $s \in Q_u$, we say that A *selects* states $\delta(s, b)$. We often use q or p to refer to states that are broadly in Q but could be in Q_e or Q_u . A transition $\delta(s, b)$ is *truly universal* if $|\delta(s, b)| > 1$. Conversely, A transition $\delta(g, b)$ is *truly existential* if $|\delta(g, b)| > 1$.

Example 2.3.1. This example provides a very simple AFA that is easy to follow for the reader. The AFA A depicted in Figure 2.1 has existential states g_0 and g_1 , universal states s_0 and s_1 , and is defined over the alphabet $\{0, 1\}$. A recognizes the language $11^*1 \cap 10^*1$, which essentially only contains the string '11'. Notice that states s_0 and g_1 are respectively truly universal and existential, while states s_1 and g_0 are not truly universal or existential.

Again, notice that an AFA can be seen as a generalization of an NFA, with universal states being the only addition. An AFA is considered an NFA if Q_u is empty. Conversely, an AFA is a *universal finite automaton* (UFA) if Q_e is empty.

Defining acceptance conditions for AFAs is far more complicated than NFAs as our definition must be inductive. A accepts the empty string ϵ starting from state q if $q \in F$. A accepts a string bw starting from state q if either:

- $q \in Q_e$ and there exists $p \in \delta(q, b)$ such that A accepts w starting from p , or
- $q \in Q_u$ and for all $p \in \delta(q, b)$, A accepts w starting from p .

Again, we give a computation tree definition of acceptance that better visualises the computation of A . The computation tree of an AFA A is defined identically to the computation trees of NFAs shown previously. However, the acceptance of the computation tree is defined inductively. The computation tree $T_{A,q,\epsilon}$ is accepting if $q \in F$. A computation tree $T_{A,q,bw}$ is said to be accepting if either

- $q \in Q_e$ and there exists $p \in \delta(q, b)$ such that $T_{A,p,w}$ is accepting, or
- $q \in Q_u$ and for all $p \in \delta(q, b)$, $T_{A,p,w}$ is accepting.

The AFA A accepts input w if $T_{A,q_0,w}$ is an accepting computation tree. Notice that unlike computation trees of NFAs, finding a series of consecutive nodes from the root of $T_{A,q_0,w}$ to an accepting leaf (q, ϵ) does not prove that A accepts w . Instead, proving that A accepts w is more complicated and requires finding a partial computation tree, as elaborated below.

Example 2.3.2. Consider the language $\text{Divisible}(m_1, m_2)$ defined over the unary alphabet $\{1\}$ for any two natural numbers m_1 and m_2 . A string w is in $\text{Divisible}(m_1, m_2)$

if the length of w is divisible by either m_1 or m_2 . Let $P(m)$ be the set of prime factors of any natural number m . We can devise an AFA A solving $\text{Divisible}(m_1, m_2)$ using $3 + |P(m_1)| + |P(m_2)|$ states.

A starts from an existential state and guesses whether the size of the input is divisible by m_1 or m_2 . Then, the branch that guessed that $|w|$ is divisible by m_1 universally selects to check that all prime factors of m_1 divide $|w|$. Similarly, the branch that guessed that $|w|$ is divisible by m_2 universally selects to check that all prime factors of m_2 divide $|w|$.

Figure 3.1 shows an example of an AFA A recognizing $\text{Divisible}(14, 15)$ with a total of 20 states. Note that state g_0 is truly existential, states s_1 and s_2 are truly universal, and the rest of the states are deterministic and thus they can be existential or universal. An NFA recognizing $\text{Divisible}(14, 15)$ would need 30 states and a DFA would need 210 states.

2.4 Existential and Universal Width of AFAs

Building on computation trees of AFAs, we seek to define existential and universal width measures, which are designed to quantify the amount of universal and existential power used in computations of an AFA.

First, we must define the process of pruning a computation tree. A *pruning* of a tree $T_{A,q,w}$ is defined inductively. If $q \in F$ and w is an empty string, the only pruning of $T_{A,q,w}$ is the tree itself. For the rest of the cases, we break down w into bw' . If $\delta(q, b) = \emptyset$, then the only pruning of the tree $T_{A,q,bw'}$ is the tree itself. Otherwise, if q is existential and there exists a state $q' \in \delta(q, b)$, a pruning of $T_{A,q,bw'}$ has a root node (q, b) that has a pruning of $T_{A,q',w'}$ as a child as well as $|\delta(q, b)| - 1$ children nodes

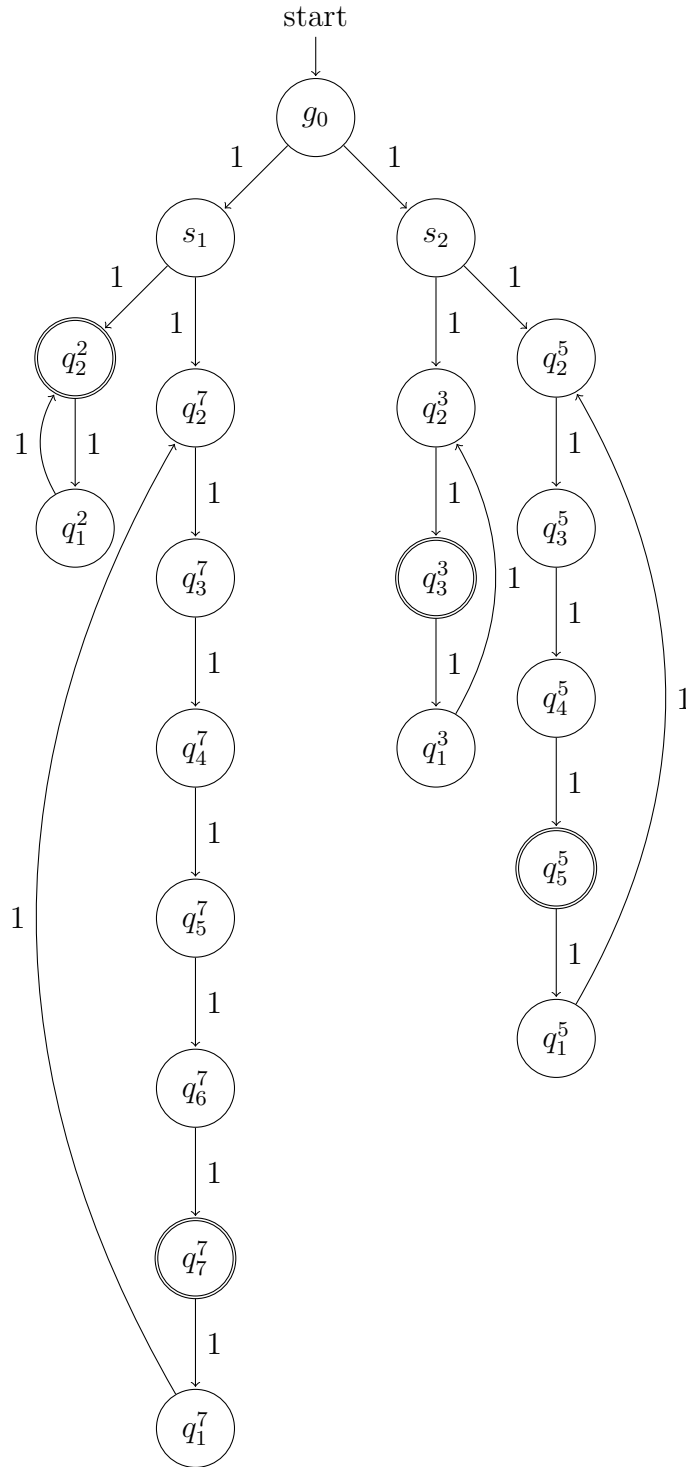


Figure 2.2: AFA A recognizing the language $\text{Divisible}(14, 15)$.

labeled as cut by the symbol ψ . Notice that there are $|\delta(q, b)|$ ways to prune the children of (q, b) , depending on which node you chose not to cut. In the case where q is universal, $T_{A,q,bw'}$ has a root node (q, b) whose children are a pruning of $T_{A,q',w'}$ for each node $q' \in \delta(q, b)$. A pruning is accepting if all its leaf nodes are accepting or labeled ψ . A pruning is said to be rejecting if it is not accepting. Notice that an accepting pruning of $T_{A,q_0,w}$ is actually a *witness* of A 's acceptance of w . This witness tree is what I referred to in chapter 1 as a witness partial computation tree.

The set of all prunings of a tree T is $\mathfrak{X}(T)$ and the set of accepting prunings is $\mathfrak{X}^{acc}(T)$. The existential width of a pruning T^p of $T_{A,q,w}$ is the number of leaves labeled by ψ and the universal width of A is the number of the rest of the leaves. We extend those definitions to the computation of A on w . The existential width of the computation of A on w , $ew(A, w)$, is the minimum existential width of any accepting pruning in $\mathfrak{X}(T_{A,q_0,w})$. Respectively, the universal width of A on w , denoted as $uw(A, w)$ is the minimum universal width of any accepting pruning in $\mathfrak{X}(T_{A,q_0,w})$. We further extend the definitions of existential and universal width for an AFA A . The existential width of A , $ew(A)$, is the supremum of all $ew(A, w)$ for $w \in L(A)$. Respectively, the universal width of A , $uw(A)$, is the supremum $uw(A, w)$ for all $w \in L(A)$.

In a very similar manner, we also define the maximal existential and universal width of A . The maximal existential width of A on input w , $ew_{max}(A, w)$, is maximum existential width of any pruning of $T_{A,q_0,w}$ and the maximum universal width of A on w , $uw_{max}(A, w)$, is the maximum universal width of any pruning of $T_{A,q_0,w}$. The maximum existential width of A , $ew_{max}(A)$, is the supremum $ew_{max}(A, w)$ for all $w \in \Sigma^*$. Respectively, the maximum universal width of A , $uw_{max}(A)$, is the supremum

$uw_{max}(A, w)$ for all $w \in \Sigma^*$.

Example 2.4.1. In Figure 2.3, we show an example of a pruned computation tree. Universal nodes of the tree are uncoloured, existential nodes are coloured in, accepting nodes have a bold outline, and cut nodes are labelled by Ψ . The pruning is accepting, has existential width of 1 and universal width of 3.

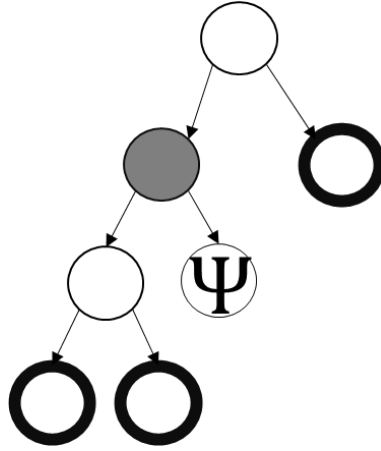


Figure 2.3: Pruned computation tree of an AFA

Lastly, we define the combined width of an AFA A on an input w , $cw(A, w)$. Intuitively, the combined width is a rough measure of the combined existential and universal width of the 'best' accepting pruned tree.

$$cw(A, w) \leq (r, s) \text{ iff } (\exists T^p \in \mathfrak{A}^{acc}(T_{A, q_0, w})) \text{ such that } ew(T^p) \leq r \text{ and } uw(T^p) \leq s.$$

Here " \leq " refers to the componentwise partial ordering of pairs of integers. Note that $cw(A, w)$ does not have a particular value because the ordering of pairs of integers is not total. Note that a pruned tree that minimizes existential width does not need

to minimize universal width and vice versa. Hence, the conditions $ew(A, w) = r$ and $uw(A, w) = s$ do not imply $cw(A, w) \leq (r, s)$, see Example 2.4.2. On the other hand, maximal width refers to all possible computation trees, and if $ew_{max}(A, w) \leq r$ and $uw_{max}(A, w) \leq s$, then any pruned computation tree of A on w has existential width at most r and universal width at most s . A similar implication does not hold for the optimal width measures and if we want to limit both values simultaneously we use combined width.

Example 2.4.2. The AFA A depicted in Figure 2.4 has existential states e_1, e_2, e_3 , and universal states u_1, u_2 . Any accepted string has an accepting pruned tree with existential width 1 and another accepting pruned tree with universal width 1, that is, $ew(A) = uw(A) = 1$. However, $cw(A, ab) \not\leq (1, 1)$ and, more generally, $cw(A)$ is not upper bounded by (r, s) for any $r, s \in \mathbb{N}$.

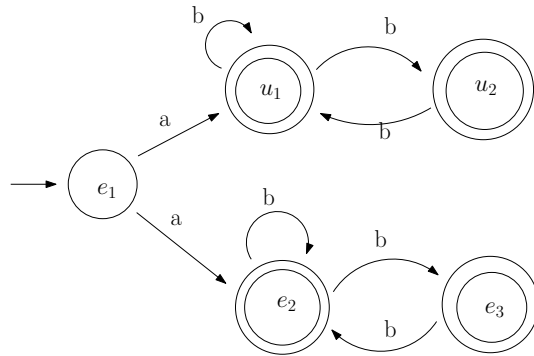


Figure 2.4: AFA A recognizing ab^*

2.5 Fooling Sets

In this research, we often use the extended fooling set technique, introduced in [3], to prove lower bounds for NFAs. We borrow our definition directly from [16]. Given a

language L , an extended fooling set S is a set of string tuples $\{(x_1, y_1), \dots, (x_n, y_n)\}$ such that any string $x_i y_i \in L$ but for all $j \neq i$, either $x_i y_j \notin L$ or $x_j y_i \notin L$.

Lemma 2.5.1. *A language L cannot be recognized by an NFA with less than n states if there exists an extended fooling set for L of size n . A set S of n tuples (x_i, y_i) is a fooling set if:*

- (i) $x_i y_i \in L$ for $1 \leq i \leq n$.
- (ii) $x_i y_j \notin L$ or $x_j y_i \notin L$ for $1 \leq i \neq j \leq n$.

As we also seek to prove lower bounds for universal finite automata (UFAs), we introduce a novel, though simple, symmetric technique, *U-fooling sets*.

Definition 2.5.1. *Let $L \subseteq \Sigma^*$ be a regular language and suppose there exists a set of pairs $S = \{(x_i, y_i) \mid 1 \leq i \leq n\}$ such that:*

- (i) $x_i y_i \notin L$ for $1 \leq i \leq n$,
- (ii) $x_i y_j \in L$ or $x_j y_i \in L$ for $1 \leq i \neq j \leq n$,

then S is a U-fooling for the language L .

Theorem 2.5.1. *If a language L has U-fooling set of cardinality n , then any UFA recognizing L has at least n states.*

Proof. Let $U = (Q, \sigma, \delta, q_0, F)$ be any UFA accepting the language L and let $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be a U-fooling set for L . Since $x_i y_i \notin L$, there is a state q_i in Q such that $q_i \in \delta(q_0, x_i)$ and $\delta(q_i, y_i)$ does not accept. Assume a fixed choice q_i , we prove that $q_i \neq q_j$ for some $i \neq j$. For the sake of contradiction, assume $q_i = q_j$ for some $i \neq j$. Then, the UFA must reject both $x_i y_j$ and $x_j y_i$. This contradicts our U-fooling set statement. □

Lastly, note that any U-fooling set for a language L is also a fooling set for $\Sigma^* - L$ and vice versa. To better understand this, notice that any string that is in a language L is, by definition, not in $\Sigma^* - L$. Because of this, any element (x_i, y_i) that is in a fooling set for L is in the language L . However, (x_i, y_i) is not in $\Sigma^* - L$.

Chapter 3

Limited Universal Width Alternating Finite Automata

We begin by exploring comparisons between limited universal width AFAs and various other models. First, we provide upper and lower bounds for comparing the state size of limited optimal and maximal universal width AFAs to NFAs. We then give better bounds for comparing limited universal width AFAs to DFAs and further improve those bounds for the special case of unary DFAs.

3.1 Finite Universal Width AFAs and NFAs

We find upper and lower bounds for comparing limited universal width AFAs to NFAs by using the classical techniques of subset construction and fooling sets. The upper bound constructions are fairly straightforward, although familiarity with subset construction will greatly aid the reader's understanding.

Recall from chapter 1 that any AFA A of size n can be simulated by an NFA B of size 2^n [6]. As this idea is extremely important to this research, we provide our own construction for the NFA B .

Proposition 3.1.1. *Any language L recognized by an AFA A with n states is recognized by a NFA B with 2^n states.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an AFA recognizing $L(A)$. We construct a NFA B simulating A as follows:

- State set P of B consists of all non-empty subsets of Q of size at most n .
- Start state of B is the state $\{q_0\}$.
- Final states of B are $P \cap 2^F$. In other words, states that are subsets of F are final.
- The transition function γ of B for a symbol $b \in \Sigma$ and a state $p = \{r_1, \dots, r_m, s_1, \dots, s_k\} \in P$, where for $i = 1, \dots, m$, $r_i \in Q_e$ and for $j = 1, \dots, k$, $s_j \in Q_u$, consists of all sets

$$\{x_1, \dots, x_m\} \cup \bigcup_{j=1}^k \delta(s_j, b)$$

where $x_i \in \delta(r_i, b)$ for $i = 1, \dots, m$, and $\delta(s_j, b) \neq \emptyset$ for $j = 1, \dots, k$.

The NFA B recognizes $L(A)$ using at most $\sum_{i=1}^n \binom{n}{i} = 2^n$ states. □

Example 3.1.1. We showcase an example of the construction shown above in Proposition 3.1.1. Consider the AFA A shown in Figure 3.1, where states s_0 and s_1 are universal while state g_0 is existential. Figure 3.2 shows a constructed NFA B using the proof of Proposition 3.1.1. Notice that since states $\{g_0, s_0\}$ and $\{g_0, s_0, s_1\}$ are not reachable from the start state $\{s_0\}$, we omit them from the graph.

The essence of the construction of B is to use subset construction to simulate the universal states of A , while using B 's own existential powers to simulate A 's

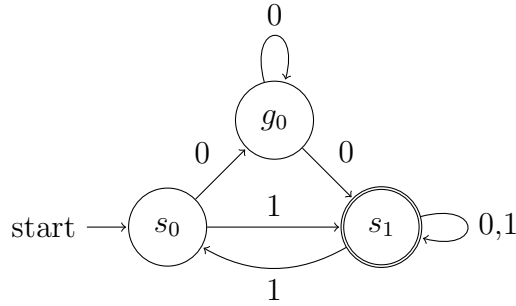


Figure 3.1: An AFA A .

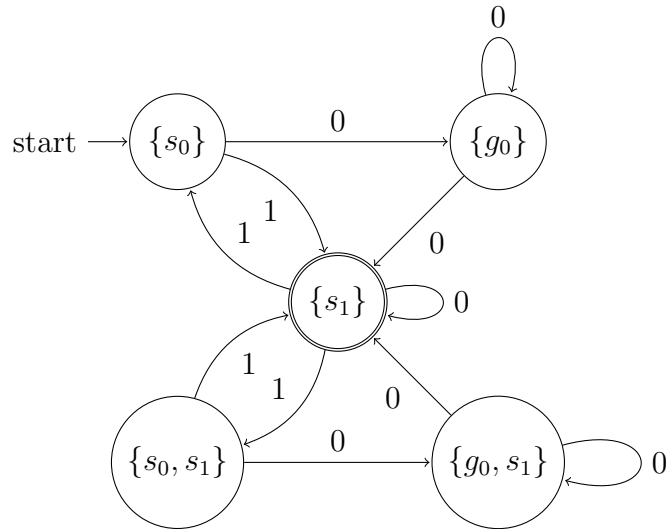


Figure 3.2: An NFA B recognizing $L(A)$.

existential states. Moreover, we are able to extend this result to DFAs using classical subset construction on B to produce a DFA C .

Proposition 3.1.2. *Any language L recognized by an AFA A with n states is recognized by a DFA C with 2^{2^n} states.*

Proof. Given any AFA A , we produce an NFA B recognizing $L(A)$ using the above construction. Afterwards, we are able to produce a DFA C also recognizing $L(A)$ using classical subset construction on B . □

Notice that this number is already extremely large for any $n = 10$ and thus providing any improvement, no matter how marginal, would make a large difference in the upper bound. Armed with our understanding of the already existing upper bounds for comparing state complexity of AFAs to NFAs and DFAs, we are ready to compare the state complexity of AFAs with limited universal width to NFAs and DFAs.

3.1.1 Upper Bounds

As shown above, the best known bound for simulating an AFA A with NFA B uses subset construction on the set of universal states to simulate universal selections while relying on its own existential powers to simulate existential guesses. We show that an AFA with limited optimal universal width can be simulated with slightly better bounds using an improved subset construction.

Theorem 3.1.1. *Any language L recognized by an AFA A with n states and $uw(A) = u < n$ can be recognized by an NFA B with $\sum_{i=1}^u \binom{n}{i}$ states.*

Proof. We construct an NFA B that recognizes $L(A)$. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an AFA recognizing $L(A)$. Let size of A be expressed by $n = |Q|$ and let $uw(A) = u < n$. We construct an NFA B simulating A as follows:

- State set P of B consists of all non-empty subsets of Q of size at most u .
- Start state of B is the state $\{q_0\}$.
- Final states of B are $P \cap 2^F$. In other words, states that are subsets of F are final.

- The transition function γ of B for a symbol $b \in \Sigma$ and a state $p = \{r_1, \dots, r_m, s_1, \dots, s_k\} \in P$, where for $i = 1, \dots, m$, $r_i \in Q_e$ and for $j = 1, \dots, k$, $s_j \in Q_u$, consists of all sets

$$\{x_1, \dots, x_m\} \cup \bigcup_{j=1}^k \delta(s_j, b)$$

where $x_i \in \delta(r_i, b)$ for $i = 1, \dots, m$, and $\delta(s_j, b) \neq \emptyset$ for $j = 1, \dots, k$ such that the cardinality of $\bigcup_{j=1}^k \delta(s_j, b)$ is at most $u - m$.

Intuitively, the constructed NFA B 'remembers' the universal selections of A by simulating at most u universal branches followed by A while the existential guesses of A are simulated using non-deterministic branching powers of B by following only one of the existential decisions taken by A . Since $uw(A) = u$, B does not need to simulate more than u universal branches of A as there must exist an accepting pruned computation tree with at most u many universal branches. \square

This construction only yields better results than 'normal' subset construction, in which we do not consider universal width, if $u < n$. In which case, the constructed NFA B has the state size $\sum_{i=1}^u \binom{n}{i}$. Notice that as u approaches n , this number becomes closer to the original bound of 2^n presented in Proposition 3.1.1. However, if u significantly smaller, the bound could be much better than the previous best bound. Similar bounds to this one are presented later on in the thesis and they follow a similar pattern.

We can construct a DFA recognizing $L(B)$, equivalently $L(A)$, by performing classic subset construction on B to obtain a DFA C . The resulting DFA C has size $2^{|B|}$.

Corollary 3.1.1. *Any language L recognized by an AFA A with n states and $uw(A) = u < n$ can be recognized by a DFA C with at most $2^{\sum_{i=1}^u \binom{n}{i}}$ many states.*

Again, those bounds are highly dependent on the values of u and n . This bound gets closer to the bound presented in Proposition 3.1.2 as u approaches n .

We can also obtain a similar bounds if we consider $uw_{max}(A)$, the maximal universal width of A , using a very similar construction with a small added improvement. The proof for this bound requires the following Lemma and follows a similar technique to proofs shown in [39].

Lemma 3.1.1. *Consider an AFA $A = (Q, \Sigma, \delta, q_0, F)$ where $uw_{max}(A)$ is finite. If a pruned computation tree T^p of A has at least two leaves, q_0 cannot label any leaf of T^p .*

Proof. For the sake of contradiction, suppose there exists an AFA A where $uw_{max}(A)$ is finite, a pruned computation tree T^p of A on a string w has at least two leaves, and at least one leaf is labelled q_0 . As the computation of A on w reaches the state q_0 at the end of w , the string w can be pumped infinitely and $uw_{max}(A, w^i) \geq i * c$, where c is a constant greater than 1 and w^i is string formed by concatenating w to itself i times. Thus, $uw_{max}(A)$ cannot be finite. \square

By using Lemma 3.1.1 and the construction used for Theorem 3.1.1, we obtain an improved bound for simulating AFAs with limited maximal universal width using NFAs.

Corollary 3.1.2. *Any language L recognized by an AFA A with n states and $uw_{max}(A) = u < n$ can be recognized by an NFA B with $1 + \sum_{i=1}^u \binom{n-1}{i}$ states.*

Furthermore, similar to Corollary 3.1.1, we can obtain a construction of a DFA C recognizing $L(B)$, equivalently $L(A)$, by applying classic subset construction on the constructed NFA B to obtain C . The constructed DFA C has $2^{|B|}$ states.

Corollary 3.1.3. *Any language L recognized by an AFA A with n states and $uw_{max}(A) = u < n$ can be recognized by a DFA C with at most $2^{1+\sum_{i=1}^u \binom{n-1}{i}}$ many states.*

These improvements only provide very marginal gains compared to the bounds presented in Theorem 3.1.1 and Corollary 3.1.1.

3.1.2 Lower Bounds

As we have established upper bounds for comparing finite universal width AFAs to NFAs, we seek to create lower bounds that are as close as possible to our upper bounds. As is commonly done in complexity theory, we produce our lower bounds by constructing languages that are provably hard for NFAs but easy for limited universal width AFAs. In our research, we were able to produce those lower bounds for AFAs with limited maximal universal width and consequently the same bounds hold for AFAs with limited optimal universal width. We show multiple examples, of varying strength. The first language, $UniCon_1^h$, is defined as follows:

- A symbol b of the alphabet Σ is defined as $b = \bigcup_{i=1}^h i \times \{0, 1, \dots, h\}$.
- A string $w = b_1, b_2, \dots, b_{|w|} \in UniCon_1^h$ if for each $b_{v+1} = \{(1, i_1), (2, i_2), \dots, (h, i_h)\}$ and $b_v = \{(1, j_1), (2, j_2), \dots, (h, j_h)\}$, for each tuple $(-, j_m) \in b_v$, there exists a tuple $(j_m, i_n \neq 0) \in b_{v+1}$. Note that in this case, j_m cannot be 0 because of alphabet limitations as there cannot be tuples of the form $(0, -)$.

In essence, a string in Σ^* represents a graph of nodes. Each symbol represents two h length columns of nodes and the edges between them. However, the alphabet only allows for each node on the left column to connect to at most one node on the right column. A connection $(j, 0)$ means that node j on the left column does not have an edge connecting it to any node on the right column. A string w is in $UniCon_1^h$ if all edges connected to the leftmost column have a path reaching the rightmost column in the graph represented by w . This problem, which may seem contrived, is inspired by the *liveliness* problem introduced in [26]. Essentially, strings in the language represent computation graphs of a UFA with some restrictions.

Example 3.1.2. Figure 3.3, taken from Kapoutsis [27], shows a visualisation of the *liveliness* problem. The problem is fairly similar to $UniCon_1^h$, except that it allows for nodes on the left column to connect to any number of nodes on the right column and asks whether there exists **any** path to the rightmost column in the input.

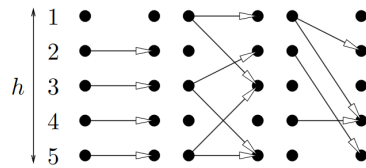


Figure 3.3: Visualising liveliness for $h = 5$ and a string of length 3.

$UniCon_1^h$ is solved by an AFA A , where the size of A is h and $uw_{max}(A) = h$. The solution is rather simple and only requires universal powers. A universally selects all the edges on the first symbol in the string, and each branch from thereon verifies that it can reach the rightmost column by remembering the last reached node in the previous symbol. A branch remembering a node j rejects if reads a symbol including

the tuples $(j, 0)$ as that signifies that it reached a dead end.

Theorem 3.1.2. *Any NFA solving $UniCon_1^h$ has at least 2^h many states.*

Proof. We employ the extended fooling sets technique introduced in chapter 2 by constructing a set S of 2^h pairs (x_i, y_i) :

- Both x_i and y_i each consist of only one symbol.
- Let $x_i = \{(1, k_1), (2, k_2), \dots, (h, k_h)\}$ and let Z_i be the i -th subset of $\{1, \dots, h\}$ in some enumeration of the subsets of $\{1, \dots, h\}$. Then, $Z_i = \bigcup_{m=1}^h \{k_m\} \mid k_m \neq 0$.
- Let $y_i = \{(1, l_1), (2, l_2), \dots, (h, l_h)\}$. For each tuple $(l, l_n) \in y_i$, $l_n = 1$ if $l \in Z_i$, otherwise $l_n = 0$.

In simpler terms, the left column in the symbol x_i connect only to the node's right column of x_i that are in subset Z_i . The left column of the symbol y_i connects all elements in Z_i to the right column of y_i . We show that S is a fooling set for $UniCon_1^h$. For each (x_i, y_i) and (x_j, y_j) such that $1 \leq i \neq j \leq 2^h$:

- (i) $x_i y_i$ is obviously $\in UniCon_1^h$, as for any tuple $(-, k_m \neq 0) \in x_i$, there exists a tuple $(k_m, 1) \in y_i$.
- (ii) $x_i y_j \notin UniCon_1^h$ or $x_j y_i \notin UniCon_1^h$. Let $x_i = \{(1, k_1), (2, k_2), \dots, (h, k_h)\}$ and $x_j = \{(1, g_1), (2, g_2), \dots, (h, g_h)\}$. There are three possible cases:
 - (a) The union of all k_m such that $k_m \neq 0$ is a subset of the union of all g_l such that $g_l \neq 0$. In this case, $x_j y_i \notin UniCon_1^h$.
 - (b) In the symmetric case, The union of all g_l such that $g_l \neq 0$ is a subset of the union of all k_m such that $k_m \neq 0$. In this case, $x_i y_j \notin UniCon_1^h$.

(c) If neither cases are true, then both $x_i y_j$ and $x_j y_i$ are not in $UniCon_1^h$.

□

Our proof above has few weak points, the alphabet of $UniCon_1^h$ is exponentially large in terms of h and the strings of our fooling set S are only of length 2. To this end, we seek to define languages over a constant sized alphabet Σ , with Σ usually including symbols 1 and 0. We define the following terms to more concisely discuss languages with an alphabet including the symbol 1.

Definition 3.1.1. *Consider an alphabet Σ where $1 \in \Sigma$. Let x be a string in Σ^* . A 1-substring of x is a substring of x that consists only of 1's. A 1-slice of x is a maximal 1-substring of x . In other words, it is a 1-substring that is surrounded by non 1 symbols. The set $S(x) \subseteq \mathbb{N}$ is the set of integers consisting of lengths of 1-slices in x .*

The terminology described above is used frequently throughout the thesis to simplify definitions of languages. The first language constructed using those definitions is $SubSet^h$, defined over the alphabet $\{0, 1, \#\}$. A string w is in $SubSet^h$ if:

- w is of the form $x\#y$, where x and y do not contain the symbol $\#$.
- There are at most h many 1-slices in x .
- $S(x)$ is a subset of $S(y)$.

In essence, a string w is in $SubSet^h$ if the set represented in binary by x is a subset of the set represented by y . $SubSet^h$ is designed to be difficult for NFAs but easy for AFAs. We can construct an AFA A recognizing $SubSet^h$ as follows:

- (i) A uses $h + 1$ states to check that prefix before $\#$ has at most h 1-slices.
- (ii) At the start of each 1-slice, computation branches universally and the new branch counts the length of the 1-slice. The counting uses h states X_1, \dots, X_h . If length of the 1-slice exceeds h , A moves into the accepting state, which moves to the end of the input and accepts.
- (iii) After the end of the 1-slice, A moves from state X_i to state Y_i , and Y_i travels until the marker $\#$.
- (iv) At marker $\#$, state Y_i becomes Z_i .
- (v) Z_i travels to the right and, nondeterministically, at start of some 1-slice starts counting down by moving into a state W_i and employing a set of states W_1, \dots, W_h to count down. Note that the nondeterministic transition from Z_i to W_i needs to be done on the symbol preceding the 1-slice.

A recognizes $SubSet^h$ using $5h + 1$ states. The maximal universal width of A is $uw_{max}(A) = h + 1$.

Theorem 3.1.3. *Any NFA solving $SubSet^h$ requires at least 2^h many states.*

Proof. Again, we use the extended fooling sets technique. We construct a set S of pairs $(x_i, \#x_i)$, where $0 \leq i < 2^h$. The string x_i is any string such that $S(x_i)$ is the i -th enumerated subset of the set $\{1, 2, \dots, h\}$.

Quite clearly, all strings $x_i\#x_i$ are in $SubSet^h$. To show that S is a fooling set, we show that for all $i \neq j$, $0 \leq i, j < 2^h$, either $x_i\#x_j \notin SubSet^h$ or $x_j\#x_i \notin SubSet^h$. There are two possible cases:

- (i) $S(x_i)$ contains an element not in $S(x_j)$. In this case, $x_i\#x_j$ is $\notin SubSet^h$.

(ii) $S(x_j)$ contains an element not in $S(x_i)$. Now, $x_j \# x_i$ is $\notin \text{SubSet}^h$.

Thus, S is a fooling set for SubSet^h and any NFA solving SubSet^h requires at least 2^h states. \square

While this result directly translates to DFAs, much stronger bounds are obtainable for DFAs.

3.2 Finite Universal Width UFAs and DFAs

We find upper and lower bounds for comparing limited universal width UFAs to DFAs, mostly by employing similar techniques and ideas to section 3.1. Again, the upper bound constructions use a specialized version of subset construction while the lower bounds rely on fooling sets.

3.2.1 Upper Bounds

To understand the interaction between finite universal width AFAs and DFAs, we seek to highlight universal aspects of AFAs while minimizing existential powers. For this reason, we compare finite universal width UFAs, the simplest version of limited universal width AFAs. First, we show an upper bound for simulating limited optimal universal width AFAs with DFAs.

Theorem 3.2.1. *Any language L recognized by an UFA A with n states and $uw(A) = u < n$ can be recognized by a DFA B with $\sum_{i=1}^u \binom{n}{i}$ states.*

Proof. We construct a DFA B that recognizes L . Let $A = (Q, \Sigma, \delta, q_0, F)$ be an UFA recognizing $L(A)$. Let size of A be expressed by $n = |Q|$ and let $uw(A) = u < n$. We construct an DFA B simulating A as follows:

- State set P of B consists of all non-empty subsets of Q of size at most u .
- Start state of B is the state $\{q_0\}$.
- Final states of B are states that consist exclusively of final states in Q
- The transition function $\gamma(p, b)$ of B for a state $p = \{q_1, \dots, q_m\} \in P$ and a symbol $b \in \Sigma$ is $\cup_1^m \delta(q_i, b)$, where the cardinality of $\cup_1^m \delta(q_i, b)$ is at most u .

Intuitively, the constructed DFA B 'remembers' the universal decisions of A by simulating at most u universal branches followed by A . Since $uw(A) = u$, B does not need to simulate more than u universal branches of A as there must exist an accepting computation tree with at most u many universal branches. Essentially, this construction is an improved subset construction. \square

We can also obtain similar bounds if we consider $uw_{max}(A)$, the maximal universal width of A , using the similar construction with a small added improvement, similar to improvements discussed in Theorem 3.1.2. In this construction, we exclude the start state of A from all states P of B , with the exception of the start state $\{q_0\}$ of A . The proof is nearly identical to the one shown for in Theorem 3.1.2.

Theorem 3.2.2. *Any language L recognized by an UFA A with n states and $uw_{max}(A) = u < n$ can be recognized by an DFA B with $1 + \sum_{i=1}^u \binom{n-1}{i}$ states.*

Improved Bound for Unary UFAs

We are able to further improve the results of Theorem 3.2.1 if we consider only UFA's with a unary alphabet by borrowing ideas from [8]. Consider a UFA $A = (Q, \{1\}, \delta, q_0, F)$ with n states and $uw(A) = h$. Since the alphabet is unary and A

operates identically on all inputs, with only the size of the input changing the computation, we are able to provide more details about the operations of A . When using the subset construction described in Theorem 3.2.1 to construct a DFA simulating A , we create a unary DFA $B = (P, \{1\}, \gamma, p_0, P_F)$ with a tail followed by a cycle, as described in Section 2.1. Using this information, we are able to obtain a tighter bound for the size of B .

As discussed in the chapter 2, any unary DFA must be comprised of two sections, the tail and the cycle. More specifically, the cycle portion of B cannot include any truly universal decision, as $uw(A)$ is finite, all truly universal decisions of A must be taken in the tail. Let the portion of the tail of B that simulates all truly universal decisions of A be p_0, p_1, \dots, p_{k-1} , where p_{k-1} makes the last truly universal decision. We call this portion the *expanding tail* of B and the rest of the tail the *shrinking tail* of B . Note that no state in the expanding tail can repeat or appear at any later point, as that would create a cycle.

We are able to obtain more specialized bounds for each portion of B . By definition, the expanding tail has k states. Furthermore, as the shrinking tail cannot include any state visited in the expanding tail, it has at most $\sum_{i=t}^h \binom{n-k}{i}$ states, where t is the number of branches of A being simulated by B after the end of the shrinking tail. Lastly, notice the cycle can only include states of size t as no truly universal decisions can be performed anymore. Specifically, B simulates t separate cycles C_i , where $1 \leq i \leq t$ of A that cannot collapse to a lower number of loops. This is because collapsing the total number of cycles cannot be re-expanded as no truly universal decisions exist in the cycle. Thus, the size of the cycle is at most the multiplication of sizes of all C_i , $\prod_{i=1}^t |C_i|$. Finally, we are able to obtain the full size

of B , $k + \sum_{i=t}^h \binom{n-k}{i} + \prod_{i=1}^t |C_i|$.

Theorem 3.2.3. *Any unary language L recognized by an UFA A with n states and $uw(A) = u < n$ can be recognized by an DFA B with $k + \sum_{i=t}^h \binom{n-k}{i} + \prod_{i=1}^t |C_i|$ states, where k is the length of the expanding tail, t cardinality of states in the cycle, and C_i is one of cycles of A simulated in the cycle portion B .*

Note that this argument also holds for the weaker case where the $uw_{max}(A) = h$, as that would imply that $uw(A) = h$.

3.2.2 Lower Bounds

To prove a lower bound for DFAs recognizing languages recognized by UFAs, we construct a language easy for UFAs but difficult for both NFAs and DFAs. Similar to $SubSet^h$, this language relies heavily on the idea of 1-slices. $SubSetHard^h$ is defined over the alphabet $\{0, 1, \#\}$. A string $w \in SubSetHard^h$ if:

- w is of the form $x\#y$, where x and y do not contain the symbol $\#$.
- $S(x) \subset S(y)$.

Notice that $SubSetHard^h$ is similar to $SubSet^h$ but with one relaxation. Namely, we remove the constraint on the number of 1-slices in x . We construct a UFA A recognizing $SubSetHard^h$ as follows:

- From the start state, A initially universally splits into states $\{s_0^0, s_1^0, \dots, s_h^0\}$.
- Each state s_i^0 verifies that $i \in S(x)$ using a set of states $\{s_i^1, s_i^2, \dots, s_i^i\}$ to count up and states $\{s_0, s_1, \dots, s_h\}$ to remember whether $i \in S(x)$. If $i \notin S(x)$, then any state s_i^j accepts by moving into an accepting state that simply moves to the end of the input and accepts.

- After reading $\#$, each state s_i moves into a state q_i^0 .
- Each state q_i^0 verifies that $i \in S(y)$ using a set of states $\{q_i^1, q_i^2, \dots, q_i^i\}$ to count up. If a 1-slice of length i is found, the state q_i^i moves into a special accepting state q_{accept} . Otherwise, any state q_i^j , for $0 < j < i$, returns to state q_i^0 if it sees a 0, essentially restarting the process of looking for a 1-slice of length i .

The UFA A recognizes $SubSetHard^h$ using $h^2 + 4h + 2$ states and $uw_{max}(A) = h$. It is clear to see that any DFA solving $SubSetHard^h$ requires at least 2^h states, as it would need to remember the subset of h present in $S(x)$. We prove this by proving a stronger statement that any NFA solving $SubSetHard^h$ has at least 2^h states.

Theorem 3.2.4. *Any NFA recognizing $SubSetHard^h$ requires at least 2^h states.*

Proof. We construct a fooling set $s = \{(x_1, \#x_1), \dots, (x_{2^h}, \#x_{2^h})\}$ for $SubSetHard^h$. Let Z_1, \dots, Z_{2^h} be an enumeration of all subsets of $\{1, \dots, h\}$. We define x_i , where $1 \leq i \leq 2^h$, as any string over $\{0, 1\}^*$ such that $S(x_i) = Z_i$.

It is clear to see that s fulfils the first statement of the fooling set as any string $x_i \# x_i$ is in $SubSetHard^h$ because $S(x_i) = S(x_i)$. As for the second statement, for any i, j such that $1 \leq i \neq j \leq 2^h$, the strings x_i and $\#x_j$ fall into one two cases:

- (i) $S(x_i) - S(x_j)$ is non-empty. In this case, $(x_j, \#x_i)$ is not in $SubSetHard^h$.
- (ii) $S(x_j) - S(x_i)$ is non-empty. In this case, $(x_i, \#x_j)$ is not in $SubSetHard^h$.

□

Corollary 3.2.1. *Any DFA recognizing $SubSetHard^h$ requires at least 2^h states.*

Chapter 4

Limited Existential Width Alternating Finite Automata

Similar to chapter 3, we explore the relations between limited existential width AFAs and a few other models. We begin by showing that finding bounds for limited optimal existential width AFAs is significantly harder than finding bounds of optimal universal width AFAs. We then show that finding bounds for limited maximum existential width AFAs is significantly easier.

4.1 Finite Existential Width AFAs and NFAs

Again, recall from chapter 1 that any AFA A of size n can be simulated by an UFA B of size 2^n [6]. Similar to chapter 3, we provide our own construction for the UFA B to better understand the current state complexity comparisons.

Proposition 4.1.1. *Any language L recognized by an AFA A with n states is recognized by a UFA B with 2^n states.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an AFA recognizing $L(A)$. We construct a UFA B simulating A as follows:

- State set P of B consists of all non-empty subsets of Q of size at most n .
- Start state of B is the state $\{q_0\}$.
- A state $p \in P$ is final if $p \cap F \neq \emptyset$. In other words, a state p is final if it includes any state in F .
- The transition function γ of B for a symbol $b \in \Sigma$ and a state p , such that $p = \{g_1, \dots, g_m, s_1, \dots, s_k\} \in P$, for $i = 1, \dots, m$, $g_i \in Q_e$, and for $j = 1, \dots, k$, $s_j \in Q_u$, consists of all states

$$\bigcup_{i=1}^k \delta(g_i, b) \cup \{x_1, \dots, x_m\}$$

where $x_j \in \delta(s_j, b)$ for $j = 1, \dots, m$.

The UFA B recognizes $L(A)$ using at most $\sum_{i=1}^n \binom{n}{i} = 2^n$ states. \square

The essence of the construction of B is to use subset construction to simulate the existential states of A , while using B 's own universal powers to simulate A 's universal states. Lastly, recall from Proposition 3.1.2 that any AFA A of size n can be simulated by a DFA C of size 2^{2^n} . Now that we have awareness of the upper bounds for comparing state complexity of general AFAs to UFAs and DFAs, we are ready to compare the state complexity of AFAs with limited existential width to UFAs and DFAs.

4.1.1 Differences Between Existential and Universal Width

Naturally, we seek to find bounds symmetric to the ones presented in chapter 3. First, we seek to establish an upper bound for the size of a UFA simulating a finite optimal

existential width AFA. Unfortunately, due to quite a few hurdles, we are unable to provide bounds anywhere close to the ones presented for limited optimal universal with AFAs.

The most notable of the differences comes from the way universal width and existential width are counted. Consider a pruning T^p of $T_{A,q_0,w}$. While the universal width of T^p counts the total number leaves in T^p , the existential width essentially counts the number of branches not explored by T^p , labeled by the ψ symbol. In essence, the universal width counts the number total number of branches in T^p , while the existential width counts the total number of unexplored branches, marked by ψ .

This causes some difficulties when trying to mirror the simulation constructions given in chapter 3 as we are not able to create constructions that simulate a singular pruning T^p in each branch. Meanwhile, the machine B constructed in the proof of Theorem 3.1.1 is able to essentially simulate one pruning T^p per branch. This is only possible because the universal width of a pruning T^p accounts for all of the properly universal power used in that pruning. However, the construction of T^p cuts all but one existential node every time an existential guess is taken, making the existential width of T^p a less indicative metric for the total amount of existential power used in the computation.

This makes us unable to achieve symmetrical bounds to the round bounds provided in chapter 3. However, we provide a workaround for bounds on AFAs with limited maximal existential width.

Theorem 4.1.1. *Any AFA A with n states and $ew_{max}(A) = e$ takes at most $k^{\frac{e}{k-1}}$ existential guesses in any computation tree, where k is the maximum number existential choices taken by A in a single step.*

Proof. Consider a pruning T_p of $T_{A,q_0,w}$ of A on a string w . Since the existential width of any pruning of A is at most e , we know that the number of non-leaf 'truly' existential nodes in T_p is at most e . This because every such node must have at least two children, one of which will be labeled by ψ , and thus each such node must contribute at least one ψ to the existential width of T_p .

Since all such nodes must be ancestors of each other in a specific ordering, we can conclude that the number of truly existential decisions of A on w in a singular branch is at most e . Assuming the worst case scenario where A always makes k guesses every time an existential guess is taken, we are able to limit the total amount of existential guesses to $k^{\frac{e}{k-1}}$. \square

In certain scenarios, we may simplify the math by slotting k to be 2, as things would get needlessly messy without providing extra insight. In such cases, the existential width of our hypothetical pruning T^p would be 2^e .

4.1.2 Upper Bounds

We provide an upper bound for simulating an AFA with limited maximal existential width using a UFA. This construction relies on the limit placed on the total number of existential branches in Theorem 4.1.1.

Theorem 4.1.2. *Any language L recognized by an AFA A with n states and $ew_{max}(A) = e$ can be recognized by a UFA B with $\sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{n}{i}$ states, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < n$.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an AFA recognizing $L(A)$. We construct a UFA B simulating A as follows:

- State set P of B consists of all non-empty subsets of Q of size at most $k^{\frac{e}{k-1}}$.
- Start state of B is the state $\{q_0\}$.
- A state $p \in P$ is final if $p \cap F \neq \emptyset$. In other words, a state p is final if it includes any state in F .
- The transition function γ of B for a symbol $b \in \Sigma$ and a state p , such that $p = \{g_1, \dots, g_m, s_1, \dots, s_k\} \in P$, for $i = 1, \dots, m$, $g_i \in Q_e$, and for $j = 1, \dots, k$, $s_j \in Q_u$, consists of all states

$$\bigcup_{i=1}^k \delta(g_i, b) \cup \{x_1, \dots, x_m\}$$

where $x_j \in \delta(s_j, b)$ for $j = 1, \dots, m$ such that the size of $\bigcup_{i=1}^k \delta(g_i, b)$ is at most $k^{\frac{e}{k-1}} - m$. We are able to have this constraint as the maximum number of branches in the computation tree of A is $k^{\frac{e}{k-1}}$ as shown by Theorem 4.1.1.

The constructed UFA B recognizes $L(A)$ using $\sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{n}{i}$ states. Similar to results in Chapter 3, this bound depends on the values of e and n , and approaches the bounds shown in Proposition 4.1.1 as $k^{\frac{e}{k-1}}$ approaches n . However, since $k^{\frac{e}{k-1}}$ approaches n exponentially quickly, this bound is less strong than the upper bounds presented in the Chapter 3. Intuitively, the UFA B 'memorizes' all the existential decisions taken by A by using states representing subsets of the states of A up to size $k^{\frac{e}{k-1}}$ while simulating the universal decisions of A using B 's own universal powers. \square

Using a similar technique as the one used in proof of Theorem 3.1.2, we can obtain an improved bound if we consider that starting state should not repeat.

Lemma 4.1.1. *Consider an AFA $A = (Q, \Sigma, \delta, q_0, F)$ where $ew_{max}(A)$ is finite. Then for any $q \in Q_e$ and $b \in \Sigma$, where $\delta(q, b) = \{p_1, \dots, p_m\}$, $m \geq 2$ the start state q_0 cannot be reachable from any of the states p_1, \dots, p_m .*

Proof. If q_0 is reachable from some $p_i, i \in \{1, \dots, m\}$, then by repeating t times, $t \in \mathbb{N}$, the cycle that goes from q_0 to q , reading b to p_i , and then reaches again q_0 , we can get a pruned computation tree with existential width at least t . \square

By using Lemma 4.1.1 and the construction used for Theorem 4.1.2, we obtain an improved bound for simulating AFAs with limited maximal existential width using UFAs. States of the constructed UFA B do not need to contain subsets of the state set of A of cardinality at least two that contain the start state q_0 .

Corollary 4.1.1. *Any language L recognized by an AFA A with n states and $ew_{max}(A) = e$ can be recognized by a UFA B with $1 + \sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{n-1}{i}$ states, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < n - 1$.*

It may not seem obvious to the reader why the construction used does not work if we use it on an AFA with limited optimal existential width. For this, we show an example of a hypothetical AFA A such that $ew(A) = e$, but the construction shown for Theorem 4.1.2 would not be sufficient to simulate A . Consider a computation of A on a string w such that $w \in L(A)$ such that A initially makes an existential guess into two branches. One branch then proceed to accept without making many more guesses, but the second branch makes an indefinitely large number of guesses. The UFA B would not be able to distinguish between the two branches as it has no prior knowledge about which branch is the 'successful' one. This would result in B rejecting w as soon as the number of existential states that need to be simulated exceeds $k^{\frac{e}{k-1}}$.

Similar to methods used in Corollary 3.1.1, we can simply use subset construction on the UFA B created in the proof of Corollary 4.1.1 to construct a DFA simulating an AFA A with limited maximal existential width.

Corollary 4.1.2. *Any language L recognized by an AFA A with n states and $ew_{max}(A) = e$ can be recognized by a DFA of size $2^{1+\sum_{i=1}^k \binom{n-1}{i}}$, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < n$.*

4.1.3 Lower Bounds

We prove lower bounds for AFAs with limited existential width using U-fooling sets. Similar to the previous chapter, we provide multiple examples, one with a large alphabet and one with a constant size alphabet. The first language, $ExCon_2^h$, is defined over the alphabet $\Sigma = \bigcup_{i=1}^h i \times \{0, 1, \dots, h\} \times \{0, 1, \dots, h\}$. A string $w = b_1, b_2, \dots, b_{|w|} \in ExCon_2^h$ if:

- The length of w is at most $\log(h)$.
- The first symbol, b_1 , is of the form $\{(1, -, -), (2, 0, 0), \dots, (i, 0, 0), \dots, (h, 0, 0)\}$. Essentially, all tuples in b_1 that are not $(1, -, -)$ must have a 0 in the second and third slot.
- For each

$$b_v = \{(1, j_1, j'_1), (2, j_2, j'_2), \dots, (h, j_h, j'_h)\} \text{ and}$$

$$b_{v+1} = \{(1, i_1, i'_1), (2, i_2, i'_2), \dots, (h, i_h, i'_h)\}$$

either there exists a tuple $(-, j_m, -) \in b_v$ and a tuple $(j_m, i_n, i'_n) \in b_{v+1}$ such that $i_n \neq 0$ or $i'_n \neq 0$ or there exists a tuple $(-, -, j'_m) \in b_v$ and a tuple $(j'_m, i_n, i'_n) \in b_{v+1}$ such that $i_n \neq 0$ or $i'_n \neq 0$.

The language $ExCon_2^h$ is very similar to $UniCon_1^h$ described for Theorem 3.1.2. Each symbol represents two h length columns of nodes and edges between them. In $UniCon_1^h$, only one edge was allowed from each node on the left column but in $ExCon_2^h$, we allow for two edges from each node on the left column. Also, the connectivity in the graph represented is existential, meaning that we only need to check that there exists a path to the rightmost column. Again, such examples are inspired from the liveness problem discussed in [26].

$ExCon_2^h$ is solved by an AFA A of size $h \log(h)$ and $ew_{max}(A) = \log(h)$. The solution is simple and only requires existential powers, A checks for paths using at most h states to remember the nodes reached so far while also maintaining a counter to ensure that the input does not exceed length $\log(h)$.

Theorem 4.1.3. *Any UFA solving $ExCon_2^h$ uses at least $\binom{h}{h/2}$ states.*

Proof. We employ the U-fooling sets technique introduced in chapter 2 by constructing a set S of $\binom{h}{h/2}$ pairs (x_i, y_i) . For each pair $x_i = b_1, b_2, \dots, b_{\log(h)-1}$ and $y_i = b_{\log(h)}$:

- x_i is a string where exactly $h/2$ of the nodes on the rightmost column are reachable such that no other string $x_j \neq x_i$ reaches the same set of nodes on the rightmost column. Note that since there are exactly $\binom{h}{h/2}$ tuples in the fooling set, this implies that each subset of nodes of size $h/2$ is reached by exactly one x_i .
- y_i is a single symbol $b_{\log(h)}$ such that all the nodes in the left column of $b_{\log(h)}$ that are **not** reached by x_i are connected to a node right column.

We show that S is a U-fooling set for $ExCon_2^h$. For each (x_i, y_i) and (x_j, y_j) such that $1 \leq i \neq j \leq \binom{h}{h/2}$:

- (i) $x_i y_i$ is obviously not in $ExCon_2^h$ as no node on the rightmost column of x_i is connected to by y_i .
- (ii) $x_i y_j$ is in $ExCon_2^h$. As x_i reaches a unique set of nodes in its rightmost column that no other x_j reaches, there must be an overlap of at least one node reached by x_i that y_j connects to, and thus $x_i y_j$ is in $ExCon_2^h$. A similar argument could be shown to show that $x_j y_i$ is in $ExCon_2^h$, but that is redundant.

□

We introduce another language with a small alphabet to prove the a similar lower bound. The language largely relies on the concept of 1-slices introduced in Definition 3.1.1. We introduce a language $ExPair^h$ over the alphabet $\Sigma = \{0, 1\}$. A string is in $ExPair^h$, where h is a power of 2, if:

- The string is of the form uw , where $|u| = \log(\frac{h}{2})$.
- There exists any two 1-slices x, y in w such that
 - the length of x and y is not more than h ,
 - the length of x is not the same as the length of y ,
 - and $|x| \equiv |y| \pmod{\frac{h}{2}}$.

$ExPair^h$ can be solved by an AFA A that initially guesses all possible 1-slice modular sizes and then verifies their existence using universal powers. A operates as follows:

- At the start A existentially branches into two states $g_{\{0,1,\dots,\frac{h}{4}-1\}}$ and $g_{\{\frac{h}{4},\frac{h}{4}+1,\dots,\frac{h}{2}-1\}}$. Similarly, each of those in turn existentially branches into states that include

half of the subset included by their predecessors and so on until we reach states $\{g_0, g_1, \dots, g_{\frac{h}{2}-1}\}$. This also counts the length of u , with each branch rejecting if the input ends before the branching is finished. We refer to this process as slow binary branching.

- Each state g_i universally branches into two states s_i^0 and $s_{i+\frac{h}{2}}^0$.
- State s_i^0 searches for a 1-slice and uses a set of states $\{s_i^0, s_i^1, \dots, s_i^i\}$ to deterministically count the number of 1's currently seen in a 1-slice. If number of 1's in the 1-slice is exactly i , A accepts by moving into an accept state. Otherwise A moves into a state s_i .
- State s_i reads the rest of the 1-slice and returns to s_i^0 once a 0 is read.

A recognizes $ExpAir^h$ using $\frac{7h}{2} + \sum_{i=1}^h i = \frac{h^2}{2} + 4h$ states and $ew_{max}(A) = \log(\frac{h}{2}) - 1$.

Lemma 4.1.2. *Any UFA solving $ExpAir^h$ requires at least $2^{h/2} - 1$ states.*

Proof. Denote $M = 2^{h/2} - 1$ and let Z_1, \dots, Z_M be an enumeration of non-empty subsets of $\{1, \dots, \frac{h}{2}\}$. We construct a U-fooling set T of M pairs (x_i, y_i) :

- x_i starts with prefix of 0's of length $\log(h)$ and $S(x_i) = Z_i$.
- $S(y_i)$ is a subset of $\{h/2 + 1, \dots, h\}$ defined as $S(y_i) = \{r + h/2 \mid 1 \leq r \leq h/2, r \notin Z_i\}$

We show that T is a U-fooling set for $ExpAir^h$. We use the definition of $S(x)$ shown earlier in Definition 3.1.1. The set T fulfills statement i of the U-fooling set described in definition 2.5.1 as for all $k \in S(x_i)$, no $l \in S(y_i)$ such that $k \equiv l \pmod{\frac{h}{2}}$ exists. Similar to our proofs earlier, T also fulfills statement ii of the U-fooling set as it falls into one of two cases:

- (i) $S(x_i) - S(x_j) \neq \emptyset$. In this case, there exists a $k \in S(x_i)$ and $l \in S(y_j)$ such that $k \equiv l \pmod{\frac{h}{2}}$ and thus $x_i y_j \in \text{Expair}^h$.
- (ii) $S(x_j) - S(x_i) \neq \emptyset$. Symmetrically to case 1, there exists a $k \in S(x_j)$ and $l \in S(y_i)$ such that $k \equiv l \pmod{\frac{h}{2}}$ and thus $x_j y_i \in \text{Expair}^h$.

□

We can further improve our results by considering a unary language. Let $k \in \mathbb{N}$ and let $m_i > k, i = 1, \dots, 2^k$ be integers that are all pairwise relatively prime. A string w is in the language $\text{NoAllDivisor}(m_1, \dots, m_{2^k})$ if the length of w is not divisible by all m_t . $\text{NoAllDivisor}(m_1, \dots, m_{2^k})$ can be solved by an AFA A operating as follows:

- Similar to the construction just above, A uses binary branching to gradually split into 2^k states $g_1^k, g_2^k \dots g_{2^k}^k$ by taking only binary existential decisions.
- From here on, each g_t^k starts a loop through states $g_t^1, \dots, g_t^k, \dots, g_t^{m_t}$ that deterministically checks whether the input is divisible by m_t , rejecting if the input ends while in state $g_t^{m_t}$ and accepting otherwise.

A is an AFA, and more specifically an NFA, recognizing $\text{NoAllDivisor}(m_1, \dots, m_{2^k})$ using $\sum_{t=1}^{2^k} m_t + 2^k - 1$ states and $ew_{\max}(A) = k$.

Lemma 4.1.3. *Any UFA solving $\text{NoAllDivisor}(m_1, \dots, m_{2^k})$ requires at least $M = \prod_{t=1}^{2^k} m_t$ states.*

Proof. We construct a U-fooling set T of M pairs (x_i, y_i) where $|x_i| = i$ and $|y_i| = M - |x_i|$. The first statement of the U-fooling set is trivially true $|x_i y_i| = M$ and thus $x_i y_i \notin \text{NoAllDivisor}(m_1, \dots, m_{2^k})$ as $|x_i y_i|$ is divisible by all m_t . The second

statement of the fooling set falls into one of two cases. For any two pairs (x_i, y_i) and (x_j, y_j) , where $i \neq j$, either $|x_i y_j| < M$ or $|x_j y_i| < M$. If $|x_i y_j| < M$ then $|x_i y_j| \in \text{NoAllDivisor}(m_1, \dots, m_{2^k})$ as all strings of length less than M cannot be divisible by all m_t as all m_t are relatively prime and M is the least common multiple of all m_t . The same reasoning applies for the case where $|x_j y_i| < M$. \square

4.2 Finite Existential Width NFAs and DFAs

In this section, we show bounds for comparing limited existential width NFAs to DFAs. We show upper and lower bounds, relying mostly on ideas presented in the previous section.

4.2.1 Upper Bounds

First, we compare DFAs to NFAs with limited maximal existential width. Using the construction shown for Theorem 4.1.2 applied on an NFA A with maximal existential width e , the resulting UFA is a DFA, giving us the following bound.

Corollary 4.2.1. *Any language L recognized by an NFA A with n states and $ew_{max}(A) = e$ can be recognized by a DFA B with $\sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{n}{i}$ states, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < n$.*

Furthermore, we are also able to improve the bound by using Lemma 4.1.1, essentially using the construction of Corollary 4.1.1 on an NFA with limited maximal existential width.

Corollary 4.2.2. *Any language L recognized by a NFA A with n states and $ew_{max}(A) = e$ can be recognized by a DFA B with $1 + \sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{n-1}{i}$ states, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < n$.*

Moreover, we are able to obtain similar results for the case where the A is only limited by the optimal existential width. However, as this requires a bit more details about NFAs with limited optimal existential width, we showcase this proof in the next section in Theorem 4.3.1.

4.2.2 Lower Bounds

To prove a lower bound for DFAs recognizing languages recognized by NFAs, we construct a language that is easy for NFAs but difficult for UFAs and DFAs. Again, we use the definitions of 1-slices and $S(x)$ from Definition 3.1.1. $ExElem^h$ is defined over the alphabet $\{0, 1, \#\}$. A string w is in $ExElem^h$, where h is a power of 2, if:

- w is of the form $px\#y$, where x and y do not contain the symbol $\#$ and p is sequence of 0's of length $\log(h)$.
- $S(x) \cap S(y)$ is non-empty.

We construct an NFA A recognizing $ExElem^h$ by first existentially guessing all possible sizes of 1-slices in x and then deterministically checking that any of them exist in y . A operates as follows:

- From the start state, A existentially splits into states $\{g_0^0, g_1^0, \dots, g_h^0\}$ over the p portion of the input using the same gradual binary splitting technique shown for Theorem 4.1.2 by using h intermediary states. This part contributes $h + 1$ states to the size of A .
- Each state g_i^0 verifies that $i \in S(x)$ using a set of states $\{g_i^1, g_i^2, \dots, g_i^i\}$ to count up and states $\{g_0, g_1, \dots, g_h\}$ to remember whether $i \in S(x)$. If $i \notin S(x)$, then

any state g_i^j halts and rejects. This part contributes $h + \sum_{i=1}^h i = h + \frac{h^2+h}{2}$ states to the size of A .

- After reading $\#$, each state g_i moves into a state q_i^0 .
- Each state q_i^0 verifies that $i \in S(y)$ using a set of states $\{q_i^1, q_i^2, \dots, q_i^i\}$ to count up. If a 1-slice of length i is found, the state q_i^i moves into a special accepting state q_{accept} . Otherwise, any state q_i^j , for $0 < j < i$, returns to state q_i^0 if it sees a 0, essentially restarting the process of looking for a 1-slice of length i . This part contributes $1 + h + \sum_{i=1}^h i = 1 + h + \frac{h^2+h}{2}$ states to the size of A .

The NFA A recognizes $ExElem^h$ using $h^2 + 4h + 2$ states and $ew_{max}(A) = \log(h)$.

We first show that any UFA solving $ExElem^h$ requires at least 2^h states using the U-fooling set technique shown in Definition 2.5.1.

Theorem 4.2.1. *Any UFA solving $ExElem^h$ requires at least 2^h states.*

Proof. We construct a U-fooling set $T = \{(px_1, \#y_1), \dots, (px_{2^h}, \#y_{2^h})\}$ for $ExElem^h$.

Let Z_1, \dots, Z_{2^h} be an enumeration of all subsets of $\{1, \dots, h\}$ and let i be an integer such that $1 \leq i \leq 2^h$. We define p, x_i, y_i as follows:

- p is a string 0's of length $\log(h)$.
- x_i is a string over $\{0, 1\}^*$ such that $S(x_i) = Z_i$.
- y_i is a string over $\{0, 1\}^*$ such that $S(y_i) = \{r \mid 1 \leq r \leq 2^h, r \notin Z_i\}$.

It is clear to see that T fulfils the first statement of the U-fooling set as any string $px_i\#y_i$ is not in $ExElem^h$ as $S(x_i) \cap S(y_i) = \emptyset$. To prove the second statement, we look at any two pairs $(px_i, \#y_i), (px_j, \#y_j)$ where $1 \leq i \neq j \leq 2^h$. The pairs of string

fulfill the statements of the second statement of the U-fooling set by falling into one two cases:

- (i) $S(x_i) - S(x_j)$ is non-empty. In this case, $(x_i, \#y_j)$ is in $ExElem^h$ as the element that is in $S(x_i)$ but not in $S(x_j)$ must also be in $S(y_j)$.
- (ii) $S(x_j) - S(x_i)$ is non-empty. In this case, $(x_j, \#y_i)$ is in $ExElem^h$ as the element that is in $S(x_j)$ but not in $S(x_i)$ must also be in $S(y_i)$.

□

As a DFA is a limitation of a UFA, the same bound then directly applies to DFAs.

Corollary 4.2.3. *Any DFA recognizing $ExElem^h$ requires at least 2^h states.*

4.3 Properties of NFAs with Limited Existential Width

In this section, we compare NFAs with limited existential width to NFAs restricted in different ways. For all the NFAs discussed below, all properly non-deterministic transitions make only two choices. This is for the sake of simplicity, although similar results can possibly be shown in cases where the NFAs have larger transitions. We use the concept of branching of NFAs defined in [15] and see it how it compares to our definitions of existential width.

First, we compare how the branching of NFA compares to the optimal existential width of the NFA. This is fairly straightforward and largely relies on the argument at the beginning of chapter 4. Recall that the branching and guessing of A are defined in Definition 2.2.2 and Definition 2.2.3 respectively. Note that since our NFA can only make binary existential choices, the math is much simpler. However, generalizing

these results to all NFAs is easily possible, but would have needlessly complicated proofs. In this case, the branching of an NFA A , $\beta(A)$, could be thought of as 2 to the power of the number of existential choices taken in a singular branch, since each choice makes at most 2 existential branches.

Lemma 4.3.1. *For any NFA A , $ew(A) \leq k$ if and only if the branching of A , $\beta(A)$, is at most 2^k .*

Proof. As all properly nondeterministic transitions of A have two choices, the existential width of A is the same as the guessing of A . $\beta(A)$ is by definition 2 to the power the guessing of A , and thus $\beta(A) \leq 2^k$. \square

We are able to make similar statements about the tree width of an NFA A , as defined in Definition 2.2.1, using more or less the same argument.

Lemma 4.3.2. *For any NFA A , if $ew_{max}(A) \leq k$ then the tree width of A , $tw(A)$, is at most 2^k .*

Proof. The maximal $tw(A)$, given that $ew_{max}(A) \leq k$, is achieved when the computation tree of A on some input w is a perfect binary tree. In this case, $ew_{max}(A) = k$ and $tw(A) = 2^k$. \square

However, when we try to prove the converse statement, we find some difficulties. Although it may seem true, proving the following statement seems to be substantially more difficult.

Claim 4.3.1. *There exists an NFA A , where $tw(A) \leq m$, but no NFA B of the same size exists such that $L(A) = L(B)$ and $ew_{max}(B) \leq \log(m)$.*

Although we are unable to prove the claim above, due to difficulty in proving state bounds of NFAs with particular existential width, we showcase a language that intuitively expresses why the statement is likely true. Consider the language

$$\text{EitherDivisor}(m, n, k) = [((a^m)^* + (a^n)^*)b]^{2^k - 1},$$

we can construct an NFA A recognizing $\text{EitherDivisor}(m, n, k)$ such that the size of A is $2^k(m + n)$, $tw(A) = 2^k + 1$, and $ew_{max}(A) = 2^k$. Note that this is not the case for any m and n , but rather the 'worst case' scenario where m and n are relatively prime. A operates by existentially guessing whether each section of a 's is divisible by m or n , while keeping track of a counter for the number of a sections seen.

Intuitively, it does not seem possible to construct an NFA B of size $2^k(m + n)$ that recognizes $\text{EitherDivisor}(m, n, k)$ and $ew_{max}(B) = k$ but we are unable to prove this statement.

Similarly, we can also compare NFAs with limited existential width to NFAs with limited maximum existential width. It may seem obvious that an NFA A with limited existential width is less restricted than an NFA B with limited maximum existential width. However, proving such a statement is also difficult.

Claim 4.3.2. *There exists an NFA A of size n and $ew(A) = k$ such that no NFA B of size n and $ew_{max}(B) = k$ recognizes $L(A)$.*

It is probably very difficult to prove the claim above. Moreover, it even seems difficult to find a candidate NFA A with $ew(A) = k$ such that $L(A)$ would not appear to have an equivalent NFA of the same size with maximum existential width k .

However, when extending the model to AFAs, we are able to provide a concrete

example of a AFA A with $ew(A) = k$ such that it does not seem possible to recognize $L(A)$ with a machine B of the same size where $ew_{\max}(B) = k$. Consider the language $\text{EitherDivisor}(m, n, k)$ discussed previously. We can construct an AFA A recognizing $\text{EitherDivisor}(m, n, k)$ where the size of A is $2^k + m + n + 2$, $ew(A) = 2^k$, and $uw(A) = 2$. Again, we are observing the worst case scenario where m and n are relatively prime as smaller AFA's do exist in when m and n are not relatively prime.

The AFA A operates by universally branching at the beginning, with one branch checking that there are 2^k a segments, while the other branch checks that the length of each a segment is divisible by m or n . It seems difficult to construct an AFA B of the same size recognizing $\text{EitherDivisor}(m, n, k)$ such that $ew_{\max}(B) = 2^k$.

However, we are able to provide an upper bound for the trade off between optimal existential width, maximal existential width, and size of NFA's. Our construction is inspired by results and techniques used in [28] and relies on counting the number of existential choices in a singular branch to make sure that the maximal existential width is limited.

Theorem 4.3.1. *Any language L recognized by an NFA A of size n and $ew(A) = e$ can be recognized by an NFA B of size $e \times n$ and $ew_{\max}(B) = e$.*

Proof. Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$ recognizing $L(A)$, we construct an NFA B recognizing $L(A)$ by simulating A . The state set P of B is the set of tuples $\{1, 2, \dots, e\} \times Q$, with the start state being $(1, q_0)$ and the final states being all states $(-, q)$ in P such that $q \in F$. The transition function γ of B is defined over two cases:

- (i) $\gamma((i, q), b) = (i + 1) \times \delta(q, b)$ for $i \in \{1, 2, \dots, e - 1\}$ and $q \in Q$.
- (ii) $\gamma((e, q), b) = (e, \delta(q, b))$ if the size of $\delta(q, b)$ is one and is undefined otherwise, halting and rejecting.

Since for $ew(A) = e$, then for every $w \in L(A)$ there must exist an accepting computation with existential width at most e . Intuitively, B counts the number of existential decisions taken and rejects if the computation makes more than e existential decisions. The NFA B has size $e \times n$ and takes at most e properly existential decisions, making $ew_{max}(B) = e$. \square

This result helps us find an upper bound for simulating NFAs with limited existential width using DFAs. Given an NFA A such that $ew(A) = e$, we simply apply the construction above to obtain an NFA B such that $ew_{max}(B) = e$, we then can apply the construction from Theorem 4.2.1 on B to obtain the DFA C .

Corollary 4.3.1. *Any language L recognized by a NFA A with n states and $ew(A) = e$ can be recognized by a DFA C with $\sum_{i=1}^{k^{\frac{e}{k-1}}} \binom{e \cdot n}{i}$ states, where k is the maximum number existential choices taken by A in a single step and $k^{\frac{e}{k-1}} < e \cdot n$.*

Chapter 5

Limited Existential and Universal Width

Alternating Finite Automata

Here, we study AFAs with limited combined width as well as AFAs with separately limited existential and universal width and show that those are two very different notions. For the following arguments, we assume that all AFAs make only binary existential and universal decisions, although similar bounds can be produced for AFAs that make at most k -ary decisions.

5.1 Limited Combined Width AFAs

If the combined width of a given AFA A is known to be bounded by a pair of integers (r, s) , we can construct an equivalent AFA B with finite maximal existential width where the number of states of B is the number of states of A multiplied by a function of r and s . For readability the below Proposition 5.1.1 is stated for a binary AFA that in each transition has at most two choices. More formally, $A = (Q, \Sigma, \delta, q_0, F)$ is a *binary AFA* if for all $q \in Q$ and $b \in \Sigma$, $|\delta(q, b)| \leq 2$. The same construction idea works for any AFA but the notations become more complicated.

The proof of the next lemma is obvious and is omitted.

Lemma 5.1.1. *Let A be a binary AFA and consider a pruned tree T^p of A and a path z from the root of T^p to a leaf.*

(i) *If the path z includes t properly existential transitions, then $ew(T^p) \geq t$.*

(ii) *If the path z includes t properly universal transitions, then $uw(T^p) \geq t + 1$.*

Proposition 5.1.1. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a binary AFA with n states where $cw(A) \leq (r, s)$, $r, s \in \mathbb{N}$. Then the language $L(A)$ has a binary AFA B with $(r+1) \cdot s \cdot n$ states where $ew_{max}(B) = r \cdot 2^{s-1}$ and $uw_{max}(B) = 2^{s-1}$.*

Proof. The AFA B simulates the computation of A and, on each branch, counts the number of properly existential and properly universal transitions encountered up to that point. Since $cw(A) \leq (r, s)$, each input accepted by A must have an accepting pruned tree with existential width r and universal width s , and we know that the computation can be terminated when, according to Lemma 5.1.1, one these values has been exceeded.

More formally, $B = (P, \Sigma, \gamma, (q_0, 0, 0), F_B)$, where $P = P_e \cup P_u$,

$$P_e = (Q_e \times \{0, \dots, r\} \times \{0, \dots, s-1\}), \quad P_u = (Q_u \times \{0, \dots, r\} \times \{0, \dots, s-1\})$$

and $F_B = (F \times \{0, \dots, r\} \times \{0, \dots, s-1\})$. The transitions of γ are defined as follows.

Consider $q \in Q$, $b \in \Sigma$ and $0 \leq i \leq r$, $0 \leq j \leq s-1$. If $\delta(q, b) = \{q_1, q_2\}$ we define:

(i) Assuming $q \in Q_e$,

$$\gamma((q, i, j), b) = \begin{cases} \{(q_1, i + 1, j), (q_2, i + 1, j)\} & \text{if } i + 1 \leq r; \\ \emptyset, & \text{otherwise.} \end{cases}$$

(ii) Assuming $q \in Q_u$,

$$\gamma((q, i, j), b) = \begin{cases} \{(q_1, i, j + 1), (q_2, i, j + 1)\} & \text{if } j + 1 \leq s - 1; \\ \emptyset, & \text{otherwise.} \end{cases}$$

If $\delta(q, b) = \{q_1\}$, $\gamma((q, i, j), b) = \{(q_1, i, j)\}$ and if $\delta(q, b) = \emptyset$, $\gamma((q, i, j), b) = \emptyset$.

The first component of the state of B directly simulates a computation of A , and the transition is existential/universal depending on the type of the transition of A . Clearly $L(B) \subseteq L(A)$. The second component of the state counts the number of properly existential transitions encountered on the path, the third component counts the number of properly universal transitions encountered and, according to (i) and (ii), the computation fails if the number of properly existential transitions exceeds r or the number of properly universal transitions exceeds $s - 1$. Since for any $w \in L(A)$, A has an accepting pruned tree $T^p \in \mathfrak{A}^{acc}(A, q_0, w)$ where $ew(T^p) \leq r$ and $uw(T^p) \leq s$, by Lemma 5.1.1, B accepts all strings accepted by A .

The rules (ii) enforce that any path from the root to a leaf in a computation tree of B can have at most $s - 1$ properly universal transitions and hence $uw_{max}(B) \leq 2^{s-1}$. The rules (i) guarantee that one path in a computation tree of B can perform at most r prunings and, together with the bound on the maximal universal width of B , this implies that $ew_{max}(B) \leq r \cdot 2^{s-1}$. \square

As illustrated in Example 2.4.2, the assumption $cw(A) \leq (r, s)$ is a stronger condition than assuming the bound separately for existential and universal width of A and the construction of Proposition 5.1.1 would not work assuming only $ew(A) \leq r$ and $uw(A) \leq s$.

5.2 Limited Universal Width AFAs

Although we are not able to simulate AFAs with separately limited existential and universal widths, we are able to create bounds for simulating AFAs with limited optimal universal width by using a technique similar that used in Theorem 4.3.1. Given an AFA A of size n such that $uw(A) = u$ and A only makes binary decisions, we construct an AFA B that directly simulates A but also counts the number of universal decisions taken so far, rejecting if the number of decisions exceeds u . Since the number of universal decisions taken by B in any branch is at most u , the maximal universal width B is limited by 2^{u-1} . Essentially, the construction is identical to the one used in Proposition 5.1.1 with the existential counters removed.

Corollary 5.2.1. *For any AFA $A = (Q, \Sigma, \delta, q_0, F)$ with n states $uw(A) = u$, $L(A)$ can be recognized by an AFA B with $u \cdot n$ states and $uw_{max}(B) \leq 2^{u-1}$*

However, the construction does not work if A has only limited existential width. In this scenario, it is possible that A has many universal decisions leading to a very large unbounded number of branches, each of which is then able to make existential choices, making the maximal existential width unbounded even with our construction. This is not the case for Corollary 5.2.1 as, by definition, only one of the existential choices contributes to the universal width each time an existential decision is made because the rest of the branches are pruned.

5.3 Separately Limited Widths

Lastly, we are able to obtain bounds for DFAs simulating AFAs with limited separate existential and universal width in a fairly simple manner. Again, we assume the AFA only makes binary existential and universal choices.

Lemma 5.3.1. *For any AFA $A = (Q, \Sigma, \delta, q_0, F)$ with n states, $ew(A) = e$, and $uw(A) = u$, $L(A)$ can be recognized by a DFA C with $\sum_{i=1}^{2^e} \binom{\sum_{j=1}^u \binom{n}{j}}{i}$ states.*

Proof. First, we use the construction in Theorem 3.1.1 to construct an NFA of B of size $\sum_{j=1}^u \binom{n}{j}$ states and $ew(B) = e$. Then, using the result in Corollary 4.3.1 on B , we are able to create a DFA C of size $\sum_{i=1}^{2^e} \binom{\sum_{j=1}^u \binom{n}{j}}{i}$. \square

The value of this bound heavily depends on the values of e , u , and n . While it may look intimidating, this bound provides significant improvements over the bound presented in Proposition 3.1.2 in cases where 2^e and u are significantly smaller than n .

Chapter 6

Summary and Conclusion

In this study, we extensively studied the relation between the state complexity of limited universal and/or existential width AFAs to various other models. First, in Chapter 3, we showed fairly close upper and lower bounds for comparing the state complexity of limited optimal and maximal universal width AFAs to NFAs and DFAs. We also showed close upper and lower bounds for comparing limited optimal and maximal universal width UFAs to DFAs.

Second, in Chapter 4, we showed rough upper and lower bounds for comparing the state complexity of limited maximal existential width AFAs to NFAs and DFAs. We also established upper and lower bounds for comparing limited optimal and maximal existential width NFAs to DFAs. Although the bounds were not as 'clean' as the bounds shown in Chapter 3, the upper and lower bounds are still valuable as they are fairly close to each other. We also went on to study how our measures of optimal and maximal existential width of NFAs compare to the already existing classic measures of tree width, branching, and guessing of NFAs.

Lastly, in Chapter 5, we compared the state complexity of AFAs with limited optimal existential and universal width to AFAs with limited maximal existential

and universal width. We also compared the state complexity of limited optimal universal width AFAs to limited maximal universal width AFAs. Lastly, we showed an improved construction for DFAs simulating AFAs with limited optimal existential and universal width. Part of the results shown in this thesis have been published in [49].

6.1 Conclusions

We had come into this study with the expectation that universal and existential width are fairly symmetrical notions. However, during the course of the study, we have found that universal and existential width to be less similar than initially expected. While the universal width accurately measured the amount of universal powers used in a pruning of a computation tree of an AFA, or a partial witness computation tree as we refer to it in the introduction, the existential width was not as accurate in measuring the existential power used in the pruning.

Instead, the existential width seemed to surprisingly have more similarity to the already existing measures of branching and guessing than initially expected. For this reason, the bounds comparing limited existential width AFAs to UFAs and DFAs were much higher than the bounds comparing limited universal width AFAs to NFAs and DFAs.

6.2 Future Work

Our work leaves a number of problems still open, a few of which are described in section 4.3. As illustrated in Claim 4.3.1, we still do not know whether an NFA A with tree width at most m exists such that no NFA B exists where B has the

same as size of A , $ew_{max}(B)$ is at most $\log(m)$, and B recognizes $L(A)$. Intuitively, this claim seems to be true as we do not expect the maximal existential width to have such a strong relation with tree width. However, showing this claim is very difficult because proving statements for nondeterministic automata of a specific size is extremely difficult.

Another interesting open problem is showcased in Claim 4.3.2. The problem is rather straight-forward and simply asks whether limited optimal existential width NFAs of size n recognize the same set of languages as limited maximal existential width NFAs of size n . Of course, intuitively, this seems to be nearly impossible as the maximal restriction is much stronger than the optimal restriction. Rather surprisingly, coming up with an example of an NFA A of size n and $ew(A) = e$ such that no NFA B of size n and $ew_{max}(A) = e$ proves to be difficult.

Lastly, we still do not know whether simulating a limited optimal existential AFA A of size n with an NFA can be done in a better way than classical subset construction. As classical subset construction yields a size blow up of 2^n for the NFA, we expect that something can be done to improve the bound with the optimal existential width limitation. Unfortunately, such an improvement has eluded us for this study, but could perhaps be possible in future studies.

Bibliography

- [1] M. Anabtawi, S. Hassan, C. Kapoutsis, and M. Zakzok. An oracle hierarchy for small one-way finite automata. In Carlos Martín-Vide, Alexander Okhotin, and Dana Shapira, editors, *Language and Automata Theory and Applications*, pages 57–69, Cham, 2019. Springer International Publishing.
- [2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- [3] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43(4):185–190, 1992.
- [4] R. Book, S. Even, S. Greibach, and G. Ott. Ambiguity in graphs and expressions. *IEEE Trans. Comput.*, 20(2):149–153, feb 1971.
- [5] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. *State Complexity of Regular Languages: Finite versus Infinite*, pages 53–73. Springer London, London, 2000.
- [6] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, jan 1981.
- [7] Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 98–108, 1976.

-
- [8] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.
- [9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.
- [10] Michael Domaratzki and Kai Salomaa. Lower bounds for the transition complexity of NFAs. *Journal of Computer and System Sciences*, 74(7):1116–1130, 2008.
- [11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994.
- [12] Viliam Geffert. An alternating hierarchy for finite automata. *Theoretical Computer Science*, 445:1–24, August 2012.
- [13] Viliam Geffert, Christos Kapoutsis, and Mohammad Zakzok. Complement for two-way alternating automata. *Acta Informatica*, 10 2021.
- [14] Jonathan Goldstine, Martin Kappes, Chandra Kintala, Hing Leung, Andreas Malcher, and Detlef Wotschke. Descriptive complexity of machines with limited resources. *J. UCS*, 8:193–234, 01 2002.
- [15] Jonathan Goldstine, C.M.R. Kintala, and Detlef Wotschke. On measuring non-determinism in regular languages. *Information and Computation*, 86(2):179–194, 1990.

-
- [16] Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In Oscar H. Ibarra and Zhe Dang, editors, *Developments in Language Theory, 10th International Conference, DLT 2006, Santa Barbara, CA, USA, June 26-29, 2006, Proceedings*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.
- [17] Yo-Sub Han, Sungmin Kim, Sang-Ki Ko, and Kai Salomaa. Existential and universal width of alternating finite automata. In *Descriptive Complexity of Formal Systems: 25th IFIP WG 1.02 International Conference, DCFS 2023, Potsdam, Germany, July 4–6, 2023, Proceedings*, page 51–64, Berlin, Heidelberg, 2023. Springer-Verlag.
- [18] Yo Sub Han, Sang Ki Ko, and Kai Salomaa. Deciding path size of nondeterministic (and input-driven) pushdown automata. *Theoretical Computer Science*, 939:170–181, January 2023. Publisher Copyright: © 2022 Elsevier B.V.
- [19] Yo-Sub Han, Arto Salomaa, and Kai Salomaa. Ambiguity, nondeterminism and state complexity of finite automata. *Acta Cybernetica*, 23:141–157, 01 2017.
- [20] Oscar H.Ibarra and B. Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In B. Monien and G. Vidal-Naquet, editors, *STACS 86*, pages 171–179, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.
- [21] Markus Holzer and Martin Kutrib. Descriptive and computational complexity of finite automata—a survey. *Information and Computation*, 209(3):456–470, 2011. Special Issue: 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

-
- [22] Juraj Hromkovič and Georg Schnitger. Ambiguity and communication. *Theory of Computing Systems*, 48(3):517–534, July 2010.
- [23] Juraj Hromkovič, Sebastian Seibert, Juhani Karhumäki, Hartmut Klauck, and Georg Schnitger. Communication complexity method for measuring nondeterminism in finite automata. *Information and Computation*, 172(2):202–217, January 2002.
- [24] Tat hung Chan and Oscar H. Ibarra. On the finite-valuedness problem for sequential machines. *Theoretical Computer Science*, 23(1):95–101, 1988.
- [25] Christos Kapoutsis and Mohammad Zakzok. Alternation in two-way finite automata. *Theoretical Computer Science*, 870:75–102, 2021. Special Issue on Implementation and Application of Automata.
- [26] Christos A Kapoutsis. Deterministic moles cannot solve liveness. *J. Autom. Lang. Comb.*, 12(1-2):215–235, 2007.
- [27] Christos A. Kapoutsis. *Size Complexity of Two-Way Finite Automata*, page 47–66. Springer Berlin Heidelberg, 2009.
- [28] Martin Kappes. Descriptive complexity of deterministic finite automata with multiple initial states. *Journal of Automata, Languages and Combinatorics*, 5:269–278, 01 2000.
- [29] C. Keeler and Kai Salomaa. Width measures of alternating finite automata. In *Descriptive Complexity of Formal Systems: 23rd IFIP WG 1.02 International Conference, DCFS 2021, Virtual Event, September 5, 2021, Proceedings*, page 88–99, Berlin, Heidelberg, 2021. Springer-Verlag.

-
- [30] Casey Keeler and Kai Salomaa. Maximal existential and universal width. *Scientific Annals of Computer Science*, XXXIII:53–77, 05 2023.
- [31] Chris Keeler and Kai Salomaa. Alternating finite automata with limited universal branching. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications*, pages 196–207, Cham, 2020. Springer International Publishing.
- [32] Chandra M. R. Kintala and Patrick C. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal on Computing*, 9(1):46–53, 1980.
- [33] Dexter Kozen. On parallelism in turing machines. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 89–97, 1976.
- [34] Ernst Leiss. Succinct representation of regular languages by boolean automata. *Theoretical Computer Science*, 13(3):323–330, 1981.
- [35] Hing Leung. Descriptive complexity of NFA of different ambiguity. *Int. J. Found. Comput. Sci.*, 16:975–984, 2005.
- [36] Hing Leung and Viktor Podolskiy. The limitedness problem on distance automata: Hashiguchi’s method revisited. *Theoretical Computer Science*, 310(1–3):147–158, January 2004.
- [37] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3), 1973.
- [38] A. Ollongren. *Definition of Programming Languages by Interpreting Automata*. A.P.I.C. studies in data processing. Academic Press, 1974.

-
- [39] Alexandros Palioudakis, Kai Salomaa, and Selim G. Akl. State complexity and limited nondeterminism. In Martin Kutrib, Nelma Moreira, and Rogério Reis, editors, *Descriptive Complexity of Formal Systems*, pages 252–265, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [40] C.H. Papadimitriou. *Computational Complexity*. Theoretical computer science. Addison-Wesley, 1994.
- [41] Michael Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 04 1959.
- [42] Bala Ravikumar and Oscar H. Ibarra. Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM Journal on Computing*, 18(6):1263–1282, 1989.
- [43] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. Technical Report UCB/ERL M78/34, EECS Department, University of California, Berkeley, May 1978.
- [44] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition, 2013.
- [45] R. E. Stearns and H. B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985.
- [46] Māris Valdats. Transition function complexity of finite automata. In *Descriptive Complexity of Formal Systems*, pages 301–313, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

-
- [47] F. Wagner. *The Virtual Finite State Machine: Executable Control Flow Specification*. Fischer-Löw, 1994.
- [48] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.
- [49] Mohammad Zakzok and Kai Salomaa. Converting finite width AFAs to non-deterministic and universal finite automata. *Theoretical Computer Science*, 996:114506, 2024.