NONDETERMINISTIC FINITE STATE COMPLEXITY

A Thesis

Presented to

The Faculty of the Department of Mathematics

University of Hawai'i, Manoa

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

By

Kayleigh Hyde

April 2013

ACKNOWLEDGMENTS

First and foremost I would like to thank Dr. Bjørn Kjos-Hanssen for teaching me about set theory and logic, and especially for suggesting the topic of this paper. Without his guidance and support, this thesis would not have been possible. Secondly, I would like to thank Jamal Hassan who always took the time to answer whatever silly question I had, Sue Hasegawa for having such a cheery smile, and Troy Ludwick for tackling all my computer problems.

Finally, I'm grateful to everyone in the mathematics department who have made my time here productive and enjoyable. I will miss you all.

ABSTRACT

We define a new measure of complexity for finite strings using nondeterministic finite automata, called *nondeterministic automatic complexity* and denoted $A_N(x)$. In this paper we prove some basic results for $A_N(x)$, give upper and lower bounds, estimate it for some specific strings, begin to classify types of strings with small complexities, and provide $A_N(x)$ for $|x| \leq 8$.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1

Introduction

We would like to develop a computable measure of complexity for finite strings over a finite alphabet. It seems intuitively obvious that some strings are more "complex" than others, however trying to measure that complexity is not straight forward. Traditionally the complexity of an object is the shortest description of it. However, the notion of "description" must be defined carefully. For example, in the Berry paradox we consider a natural number described as "the least integer not namable in fewer than nineteen syllables." If this number does exist, we have a contradiction since the above description only has eighteen syllables. If no such number exists, then all natural numbers can be described in fewer than nineteen syllables [4].

In 1965 Andrey Kolmogorov defined the algorithmic complexity of an object to be the length of the shortest binary computer program that describes it [5]. Capitalizing on the simplicity of the universal Turing machine, Gregory Chaitin along with Kolmogorov independently described Kolmogorov complexity, $C(x)$, the measure of the complexity of a string x, as the size of the shortest pair (T, y) =(Turing machine description, input) such that T on input y outputs x [11].

Despite its usefulness in proving results comparable to Gödel's Incompleteness Theorem and Turing's halting problem, $C(x)$ has considerable deficiencies [8]:

- It is not computable! It is known that "$C(x) < n$" is Turing-Recognizable, but "$C(x) \geq n$" is not.
- There is no effective procedure for finding the compression pair (T, y).
- The complexity is defined in a machine-independent way only up to an additive constant.

Thus $C(xx) = C(x) + \mathcal{O}(1)$, with the constant depending on the model of universal Turing machine. Hence, we cannot ask if $C(xx) > C(x)$ for any, most, or all strings x.

Ideally, we would like to define a measure of complexity without these deficiencies. To do so we consider replacing the Turing machine with a less powerful model. For example, many have examined replacing it with a context-free grammar (CFG) to get a measure of complexity associated with word chains [6, 1]. Another measure was defined by Jeffrey Shallit and Ming-wei Wang using a deterministic finite automaton [11].

Motivated by Shallit and Wang, in this paper we will replace the Turing machine with a nondeterministic finite automata, or NFA. Nondeterminism is a generalization of determinism, so every deterministic finite automaton is also a nondeterministic finite automaton. In addition, NFAs are often smaller and easier to understand than their DFA counterparts, and are especially useful in proving closure properties of regular languages [12].

An NFA is typically described using a directed graph, and is considered a type of finite state machine. Each vertex of the graph represents a state, and edges represent possible transitions. An input string (of finite length) is read by the machine. The initial state is designated by an inward arrow that has no source vertex. The machine starts in this state and reads the first symbol of the input string. Based on its value, the NFA makes appropriate transitions.

From a state in an NFA, there may be any number of outgoing edges (including zero) that represent the response to a single symbol. There are also edges designated with a special ϵ symbol. If a state has an outgoing ϵ edge, the state may immediately transition along the edge without reading another symbol. The nondeterminism arises from the fact that there are multiple choices for possible next states.

One way to interpret what the NFA does when there are multiple choices is to think of the machine cloning itself and one copy runs each choice. If there are no outgoing edges for a certain combination of state and input, then the clone dies. Any states that are depicted with a double boundary are called *accept* states. When the input string ends, the NFA is said to accept the input string if there exits at least one clone in which the final machine state is an accept state.

Formally, an NFA is a 5-tuple $(Q, \Sigma, \delta, q_1, F)$, where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma_\epsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
- $q_1 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accept states.

Note that $\mathcal{P}(Q)$ is the power set of Q, ϵ is the empty string, and $\Sigma_\epsilon = \Sigma \cup \{\, \epsilon\}$.

CHAPTER 2

Basic Results

Let M be a nondeterministic finite automaton, having q states and no $\epsilon-$transitions. If there is exactly one path through M of length $|x|$ leading to an accept state, and x is the string read along that path, then we say that $A_N(x) \leq q$. Such an NFA is said to witness the complexity of x being no more than q. To wit:

Definition 2.1. *Let $|\Sigma| = k \geq 2$, and $x \in \Sigma^*$ with $|x| = n$. Define $A_N(x)$ to be the smallest number of states in any NFA M such that there is exactly one path through M of length n leading to an accept state and $L(M) \cap \Sigma^n = \{x\}$.*

Clearly $A_N(x) \leq |x|$, since we can uniquely accept any string of length $|x|$ with a chain of $|x|$ states that loop back to the start state. We can test each NFA with $|x|$ states to see if it uniquely accepts x by brute force, so it follows that $A_N(x)$ is computable. We would still like to know if $A_N(x)$ is computable in polynomial time in $|x|$.

We can use $A_N(x)$ as a data compression technique in the following way:

Theorem 2.2. *Given a description of a NFA M which uniquely accepts x, and $|x| = n$, we can efficiently recover x.*

Proof. We want to be able to determine x from M and we want the time it takes to be polynomial in the description size of M and n.

Let $M = (Q, \Sigma, \delta, q_1, F)$ and $Q = \{q_1, q_2, \ldots, q_r\}$ with $|Q| = r$. Consider a directed graph $G = (V, E)$ with vertex set defined as follows:

$$V = \{P_{i,j} : 1 \leq i \leq r, 0 \leq j \leq n\}.$$

So $P_{i,j}$ will be the i^{th} state reached after reading the j^{th} symbol. Place a directed edge from $P_{i,j}$ to $P_{k,l}$, denoted $(P_{i,j}, P_{k,l})$, labeled a if $\delta(q_i, a) = q_k$ and $l = j + 1$. Note that G is acyclic, because the edge $(P_{i,j}, P_{k,l})$ with $j \geq l$ will fail the $l = j + 1$ requirement.

Because M uniquely accepts x, there exits a single t with $q_t \in F$ such that there is exactly one path from $P_{1,0}$ to $P_{t,n}$. We can find this path with a depth-first search in $\mathcal{O}(|V| + |E|)$ time [10]. There are $(n + 1) \cdot |Q|$ vertices in G, and G can potentially have $|\Sigma| \cdot |V|$ edges from each vertex. Thus, the maximum number of edges in G is $|\Sigma| \cdot |V|^2$. Therefore, the time complexity for the DFS is

$$\begin{aligned}
\mathcal{O}(|V| + |E|) &= \mathcal{O}\left((n + 1)|Q| + |\Sigma| \cdot (n + 1)^2 |Q|^2\right) \\
&= \mathcal{O}\left(n^2 |Q|^2\right) \\
&= \mathcal{O}\left(n^4\right)
\end{aligned}$$

□

Theorem 2.3. *Given an NFA M with r states and a string x of length $n \geq 1$, we can determine in $\mathcal{O}(n + r^3 log_2(n))$ steps whether M uniquely accepts x.*

Proof. Let $M = (Q, \Sigma, \delta, q_1, F)$, with $Q = (q_1, q_2, \ldots, q_r)$. We can determine if M accepts x by simply simulating it on x. This can be done in $\mathcal{O}(n)$ time.

Let A be the $r \times r$ adjacency matrix for M such that

$$A_{r,r} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,r} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r,1} & a_{r,2} & \cdots & a_{r,r} \end{pmatrix}$$

where $a_{i,j}$ is the number of $b \in \Sigma_\epsilon$ such that $\delta(q_i, b) = q_j$. Let the kth matrix product, A^k, be defined as follows:

$$A^k(i,j) = \sum_{s=1}^{r} A^{k-1}(i,s)A(s,j) \quad \textit{where}$$

$$A^1(i,j) = A(i,j).$$

Then $A^k(i,j)$ is the number of non-simple paths from q_i to q_j, containing k edges [9, 7]. If we let $q_t \in F$, then M accepts x and $A^n(q_1, q_t) = 1$, if and only if M uniquely accepts x. Thus it suffices to compute A^n efficiently.

A^n can be computed with the "binary method" of exponentiation [2]. Since n has $\lfloor log_2(n) \rfloor$ binary digits, we will need $\mathcal{O}\left(log_2(n)\right)$ matrix multiplications and squarings each taking $\mathcal{O}\left(r^2(2r-1)\right) = \mathcal{O}(r^3)$ steps [3]. Hence, the total steps needed to to determine if M uniquely accepts x is $\mathcal{O}(n + r^3 log_2(n))$. □

Theorem 2.4. *$A_N(xy) \geq A_N(x)$ for all strings x.*

Proof. Consider an NFA $M = (Q, \Sigma, \delta, q_1, F)$ witnessing $A_N(xy)$. Then there must be one and only one path of length $|xy|$ from q_1 to a state of F. Label this path xy, and let $\delta(q_1, x) = q$.

Now construct a new NFA $M^{'} = (Q, \Sigma, \delta, q_1, \{q\})$. Suppose that there exists another string $w \neq x$, $|w| = |x|$, such that $\delta(q_1, w) = q$. Then it must be that $\delta(q_1, wy) \in F$, and thus M will accept another string of length $|xy|$. Therefore $M^{'}$, which is no bigger than M, must uniquely accept x.

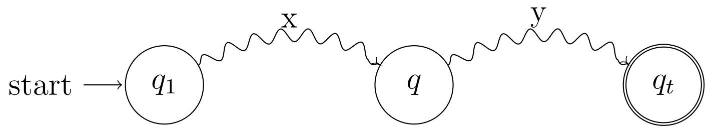


Figure 2.1: NFA M witnessing $A_N(xy)$.

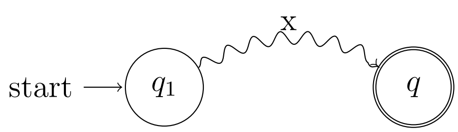


Figure 2.2: NFA M' witnessing $A_N(x)$.

$\square$

CHAPTER 3

Bounds on $A_N(x)$

3.1 Upper Bound

In this section we improve our upper bound on $A_N(x)$.

Theorem 3.1. *Let* $|\Sigma| = k \geq 2$ *be fixed and suppose* $x \in \Sigma^n$. *Then* $A_N(x) \leq \frac{n}{2} + 1$.

Proof. We will prove this upper bound by cases. Let $x = x_1 x_2 \ldots x_n$.

(Case 1: $n = 2m + 1$) Consider an NFA $M = (Q, \Sigma, \delta, q_1, \{q_1\})$ with $Q = \{q_1, q_2, \ldots, q_{m+1}\}$ and δ defined as follows:

- $\delta(q_1, x_1) = q_2$
- for $2 \leq i \leq m$, $\quad \delta(q_i, x_i) = q_{i+1}$,

  and $\delta\left(q_i, x_{(n-m+2)}\right) = q_{i-1}$
- $\delta(q_{m+1}, x_{m+1}) = q_{m+1}$
- $\delta(q_{m+1}, x_{m+2}) = q_m$

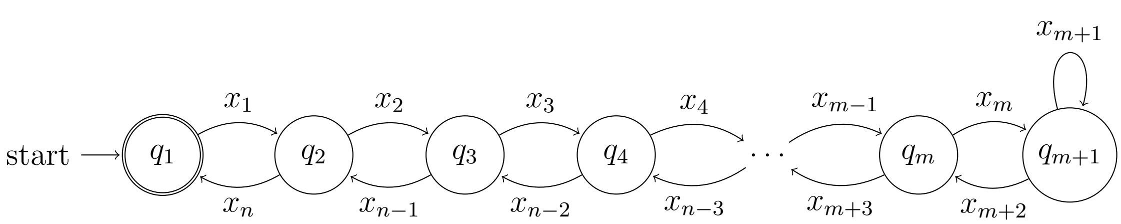


Figure 3.1: An NFA uniquely accepting $x = x_1 x_2 x_3 x_4 \ldots x_n$, $n = 2m + 1$

Let the numerical index of the state q_i, and of the symbol s_i and be i. In such an NFA it is impossible for the parity of the numerical index of the state and symbol read to be the same without going through the self loop in the q_{m+1} state first. Thus, in order to reach the accept state, q_1, after reading $|x| = 2m+1$ symbols, we must go through the self loop in q_{m+1} first. It takes m symbols to reach q_{m+1}, one symbol to go through the self loop, and m more symbols to get back to q_1. Therefore, the only string of length $|x| = 2m + 1$ that M will accept is x.

(Case 2: n = 2m) Similarly, consider an NFA $M' = (Q, \Sigma, \delta, q_1, \{q_2\})$ with $Q = \{q_1, q_2, \ldots, q_{m+1}\}$ and δ defined as follows:

- $\delta(q_1, x_1) = q_2$
- $\delta(q_2, x_2) = q_3$
- $\delta(q_{m+1}, x_{m+1}) = q_{m+1}$
- $\delta(q_{m+1}, x_{m+2}) = q_m$
- for $3 \leq i \leq m$, $\delta(q_i, x_i) = q_{i+1}$,

  and $\delta\left(q_i, x_{(n-m+3)}\right) = q_{i-1}$

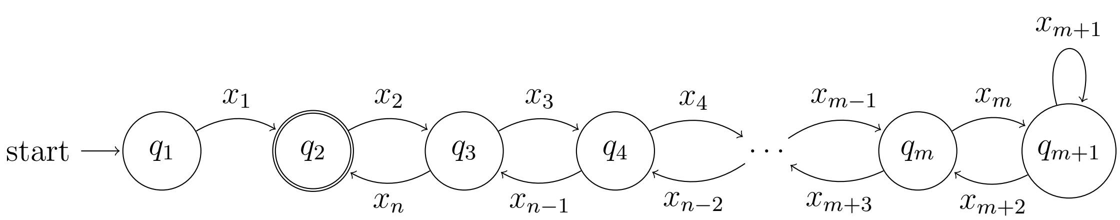


Figure 3.2: An NFA uniquely accepting $x = x_1x_2x_3x_4 \ldots x_n$, $n = 2m$

In order to reach the accept state, q_2, after reading $|x| = 2m$ symbols, we must go through the self loop in q_{m+1} first. It takes m symbols to reach q_{m+1}, one symbol to go through the self loop, and $(m - 1)$ more symbols to get back to q_2. Therefore, the only string of length $m + 1 + (m - 1) = 2m = |x|$ that M will accept is x. Hence $A_N(x) \leq \frac{n}{2} + 1$. $\square$

3.1.1 Examples

Figures 3.3 – 3.6 show NFAs witnessing the upper bound of $A_N(x)$ for a given x.

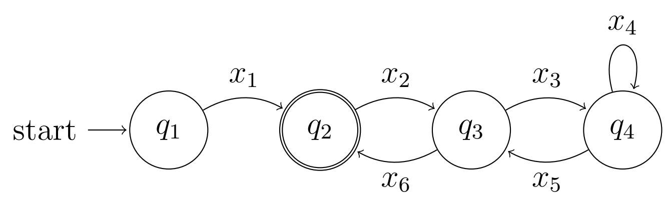


Figure 3.3: An NFA uniquely accepting $x = x_1x_2x_3x_4x_5x_6$. $A_N(x) = 4$.

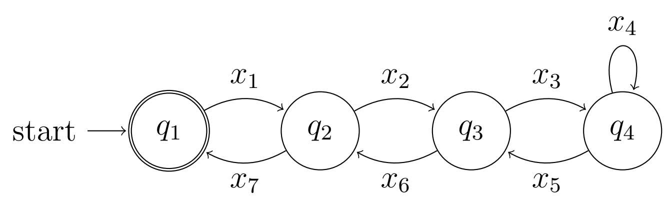


Figure 3.4: An NFA uniquely accepting $x = x_1x_2x_3x_4x_5x_6x_7$. $A_N(x) = 4$.

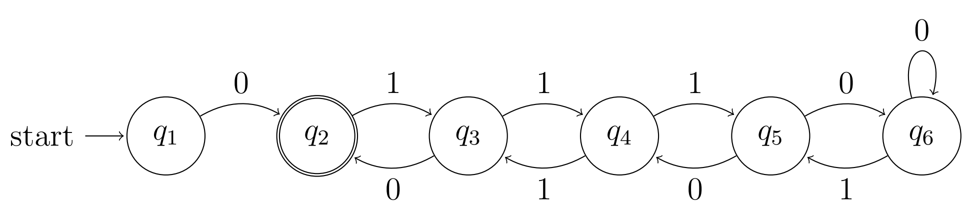


Figure 3.5: An NFA uniquely accepting $x = 0111001010$, $|x| = 10$, $A_N(x) = 6$.

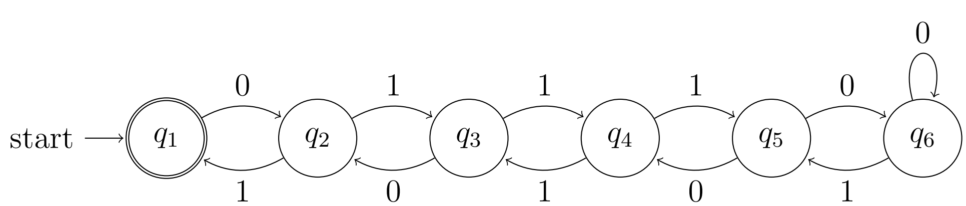


Figure 3.6: An NFA uniquely accepting $x = 01110010101$, $|x| = 11$, $A_N(x) = 6$.

3.2 Lower Bounds

We begin this section by proving some results on connected NFAs. A "connected" NFA is one where there exists at least one path from the start state into every other state.

Lemma 3.2. *Let* $Aut_q = \{$*all NFA with* q *states*$\}$ *and* $Conn_q = \{$*all NFA with* q *connected states*$\}$. *Then* $\sum_{k \leq q} |Conn_k| \leq |Aut_q|$.

Proof. For $k \leq q$, let φ be a function which adds $(q-k)$ non-connected states to an NFA of size k. For $i \neq j$, $Conn_i \cap Conn_j = \emptyset$ by definition. Hence $\varphi(Conn_i) \neq \varphi(Conn_j)$.

Thus, $\varphi : \bigcup_{k \leq q} Conn_k \rightarrow Aut_q$ is an injective function, and

$$\sum_{k \leq q} |Conn_k| \leq |Aut_q|$$

□

Lemma 3.3. *If there exists a NFA* M *with* $|M| \leq q$ *such that the only string of length* $n = |x|$ *that* M *accepts is* x*, then there exits a connected NFA* K*,* $|K| \leq q$ *that also accepts* x*.*

Proof. If M accepts x, then there must be some connected path from the start state to the final state. Let K be the initially connected components of M, and we have $|K| \leq |M| \leq q$, K completely connected, and K uniquely accepting x. □

Theorem 3.4. *Let $|\Sigma| = k \geq 2$, and let $0 < \epsilon < 1$ be fixed. Then,*

$$A_N(x) \geq \sqrt{\tfrac{1}{k} \cdot log_2\left(\tfrac{k^{\epsilon n}}{n}\right)}$$

for all strings $x \in \Sigma^n$, with at most $k^{\epsilon n}$ exceptions.

Proof. For a NFA with q states, there are 2^q subsets of states. Since there could potentially be transitions (for each symbol) from each state into any of the others, there are at most $(2^q)^{qk} \cdot q = (2^{q^2k}) \cdot q$ connected automata with $\leq q$ states and exactly one final state. Some of these automata uniquely accept at most one string of length n. Let $Simp_q^n = |\{x : |x| = n \ \& \ A_N(x) \leq q\}|$. Thus if

$$Simp_q^n \leq (2^{q^2k}) \cdot q < k^{\epsilon n}$$

then at most $k^{\epsilon n}$ different strings of length n can be represented. Now,

$$\begin{array}{rll}
 & q & < \sqrt{\dfrac{1}{k} \cdot log_2\left(\dfrac{k^{\epsilon n}}{n}\right)} \\
\iff & q^2 & < \dfrac{1}{k} \cdot log_2\left(\dfrac{k^{\epsilon n}}{n}\right) \\
\iff & q^2 k & < log_2\left(\dfrac{k^{\epsilon n}}{n}\right) \\
\iff & 2^{q^2k} & < \dfrac{k^{\epsilon n}}{n} \\
\iff & n \cdot 2^{q^2k} & < k^{\epsilon n} \\
\Rightarrow & q \cdot 2^{q^2k} & < k^{\epsilon n}
\end{array}$$

since we may assume $q \leq n$. Therefore, at most $k^{\epsilon n}$ strings of length n will have $A_N(x) < \sqrt{\tfrac{1}{k} \cdot log_2\left(\tfrac{k^{\epsilon n}}{n}\right)}$. □

Lemma 3.5. *Given $0 < \epsilon < 1$, $k \geq 2$, $b > 0$ and $b \neq 1$, fixed with $n > 1$, we have $2^{\epsilon n} > n > b^{\left(\frac{nk}{\epsilon n - log_2(n)}\right)}$ for all but finitely many n.*

Proof. First we'll show $2^{\epsilon n} > n$ for all but finitely many n. Note:

$$2^{\epsilon n} = e^{ln(2^{\epsilon n})} = e^{\epsilon n \cdot ln(2)}.$$

Now using the Taylor expansion of e^x we have,

$$e^{\epsilon n \cdot ln(2)} \geq 1 + \epsilon n \cdot ln(2) + \frac{(\epsilon n \cdot ln(2))^2}{2} = 1 + \epsilon n \cdot ln(2) + \frac{\epsilon^2 ln^2(2)}{2} \cdot n^2.$$

Thus, there exists a finite N such that $\forall n > N$ the n^2 term will dominate and

$$1 + \epsilon n \cdot ln(2) + \frac{\epsilon^2 ln^2(2)}{2} \cdot n^2 > n.$$

Hence $2^{\epsilon n} > n$ for all but finitely many n.

It remains to show $n > b^{\left(\frac{nk}{\epsilon n - log_2(n)}\right)}$ for all but finitely many n.

$$b^{\left(\frac{nk}{\epsilon n - log_2(n)}\right)} < b^{\left(\frac{nk}{\epsilon n - n}\right)} = b^{\left(\frac{k}{\epsilon - 1}\right)}$$

Since $b^{\left(\frac{k}{\epsilon-1}\right)}$ is a constant, we have $n > b^{\left(\frac{k}{\epsilon-1}\right)}$ for all but the finitely many n. $\square$

Theorem 3.6. *Let $b > 0$ and $b \neq 1$, fixed. Then $A_N(x) > \sqrt{\frac{|x|}{log_b|x|}}$ for all but finitely many x.*

Proof. Let $|\Sigma| = k \geq 2$, and $0 < \epsilon < 1$ be fixed, and suppose $|x| = n > 1$. Then by Lemma 3.5 we have $2^{\epsilon n} > n > b^{\left(\frac{nk}{\epsilon n - log_2(n)}\right)}$ for all but finitely many n. We know, by Theorem 3.4 and the change of base formula of logarithms, that for any base b, $A_N \geq \sqrt{\frac{1}{k} \cdot log_2\left(\frac{k^{\epsilon n}}{n}\right)} = \sqrt{\frac{log_b\left(\frac{k^{\epsilon n}}{n}\right)}{log_b(2^k)}}$ with at most $k^{\epsilon n}$ exceptions. Thus, it suffices to show

$$\sqrt{\frac{log_b\left(\frac{k^{\epsilon n}}{n}\right)}{log_b(2^k)}} > \sqrt{\frac{n}{log_b(n)}}.$$

Note: $2^{\epsilon n} > n \iff log_2(2^{\epsilon n}) > log_2(n) \iff \epsilon n > log_2(n) \iff \epsilon n - log_2(n) > 0$.

Now consider,

$$\begin{aligned}
& n && > b^{\left(\frac{nk}{\epsilon n - log_2(n)}\right)} \\
\iff\quad & log_b(n) && > \left(\frac{nk}{\epsilon n - log_2(n)}\right) \\
\iff\quad & log_b(n) \cdot (\,\epsilon n - log_2(n)) && > nk \\
\iff\quad & \epsilon n \cdot log_b(n) - log_b(n) \cdot log_2(n) && > nk \\
\iff\quad & \epsilon n \cdot log_b(n) - nk && > log_b(n) \cdot log_2(n) \\
\iff\quad & n(\,\epsilon \cdot log_b(n) - k) && > log_b(n) \cdot log_2(n) \\
\iff\quad & n(\,\epsilon \cdot log_b(n) - k) && > log_2\left(n^{log_b(n)}\right) \\
\iff\quad & 2^{n(\,\epsilon \cdot log_b(n) - k)} && > n^{log_b(n)} \\
\iff\quad & \frac{2^{\,\epsilon n \cdot log_b(n)}}{2^{nk}} && > n^{log_b(n)} \\
\iff\quad & \left(\frac{2^{\,\epsilon n \cdot log_b(n)}}{n^{log_b(n)}}\right) && > 2^{nk} \\
\iff\quad & \left(\frac{k^{\,\epsilon n \cdot log_b(n)}}{n^{log_b(n)}}\right) && > 2^{nk} \\
\iff\quad & \left(\frac{k^{\,\epsilon n}}{n}\right)^{log_b(n)} && > 2^{nk} \\
\iff\quad & log_b\left(\left(\frac{k^{\,\epsilon n}}{n}\right)^{log_b(n)}\right) && > log_b(2^{nk}) \\
\iff\quad & log_b(n) \cdot log_b\left(\frac{k^{\,\epsilon n}}{n}\right) && > n \cdot log_b(2^k) \\
\iff\quad & \frac{log_b\left(\frac{k^{\,\epsilon n}}{n}\right)}{log_b(2^k)} && > \frac{n}{log_b(n)} \\
\iff\quad & \sqrt{\frac{log_b\left(\frac{k^{\,\epsilon n}}{n}\right)}{log_b(2^k)}} && > \sqrt{\frac{n}{log_b(n)}}
\end{aligned}$$

$\square$

CHAPTER 4

Some Specific Examples

In this section we determine $A_N(x)$ for some specific strings. When we refer to a string of the form $x = (x_1x_2\ldots x_k)^n$, we mean the pattern $x_1x_2\ldots x_k$ is repeated n times.

Example 4.1. *Let $x = 0^m1^n$. Then $A_N(x) \leq min\{m, n\} + 1$.*

Without loss of generality, let $m < n$ and consider the witnessing NFA:

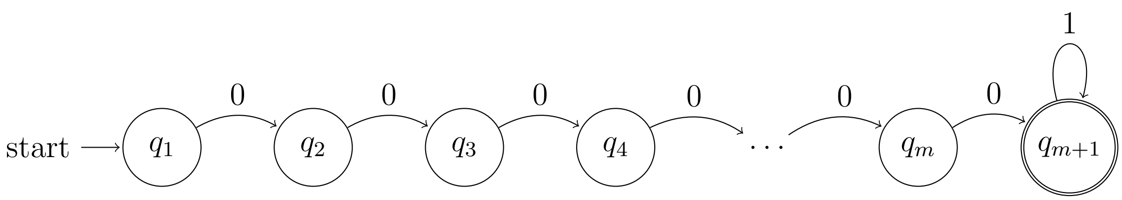


Figure 4.1: An NFA uniquely accepting 0^m1^n.

Example 4.2. *Let $x = (01)^n$, then $A_N(x) = 2$.*

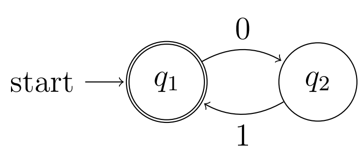


Figure 4.2: An NFA uniquely accepting $(01)^n$.

The following example will generalize this notion.

Example 4.3. *Let $x = (x_1, x_2, x_3, \ldots, x_m)^n$. Then $A_N(x) \le m$.*

Figure 4.3 is an NFA which only accepts strings with length $m \cdot k$, $k \in \mathbb{N}$. Thus, by labeling the transitions with the appropriate symbols from x, we have a witnessing automaton for $A_N(x)$.

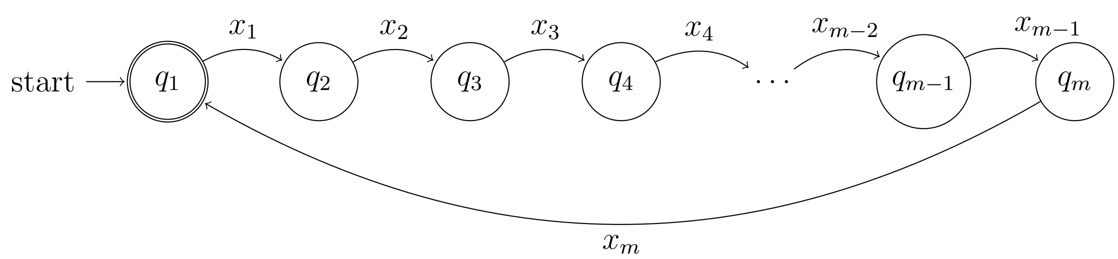


Figure 4.3: An NFA uniquely accepting $(x_1, x_2, x_3, \ldots, x_m)^n$.

4.1 Small Witnessing Automata

We consider an "abstract" witnessing NFA to be an NFA with unlabeled transitions such that if we labeled them with the appropriate symbols from x, the NFA will witness the complexity of x.

Definition 4.4. *Let $q \le \frac{n}{2} + 1$ and let $W_{q,n}$ be the set of abstract NFAs which, if labeled with the appropriate symbols from x, will witnesses $A_N(x) = q$ where $|x| = n$. Then define $N(q, n)$ to be the minimum cardinality of $S \subseteq W_{q,n}$ such that if $|x| = n$ and $A_N(x) = q$, then there is one NFA in S which witnesses the complexity.*

By examining these automata we can classify what "types" of strings, with $|x| = n$, can have $A_N(x) \le q$. For example, in Table 4.1 we see that the only strings with $A_N(x) = 1$ are those of the form $x = (x_1)^n$, where $|x| = n$.

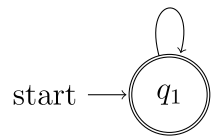
<table>
<tr>
<td>$N(1,n) = 1$, $\forall n$</td>
<td>
</td>
</tr>
</table>

Table 4.1: Abstract NFA witnessing $A_N(x) = 1$.

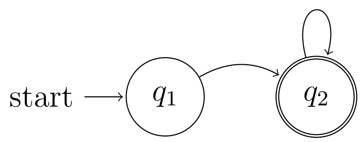


Table 4.2: $N(2,2) = 1$ abstract NFA witnessing $A_N(x) \leq 2, |x| = 2$.

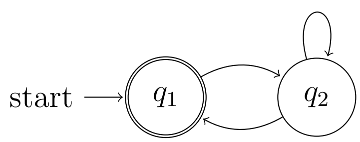
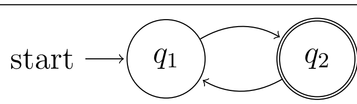
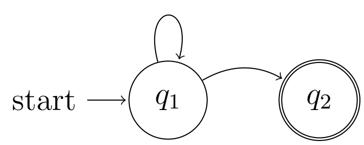
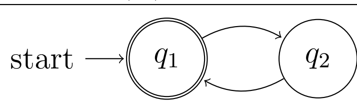


Table 4.3: $N(2,3) = 1$ abstract NFA witnessing $A_N(x) \leq 2, |x| = 3$.

<table>
<tr>
<td>

Only for $|x| = 2m + 1$</td>
<td>
</td>
</tr>
<tr>
<td>

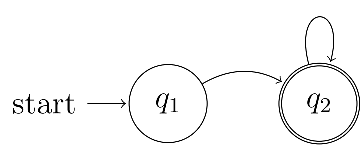
Only for $|x| = 2m$</td>
<td>
</td>
</tr>
</table>

Table 4.4: $N(2,n) = 3$ abstract NFAs witnessing $A_N(x) \leq 2, |x| > 3$.

In Tables 4.3, 4.5, and 4.6 we only see the result of our upper bound, however Table 4.4 allows us to classify all the strings with complexity at most 2. They will be strings of the form:

- $x = (x_1x_2)^m$
- $x = (x_1)^m x_2$
- $x = x_1(x_2)^m$.

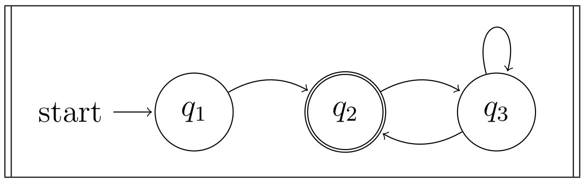


Table 4.5: $N(3, 4) = 1$ abstract NFA witnessing $A_N(x) \leq 3, |x| = 4$.

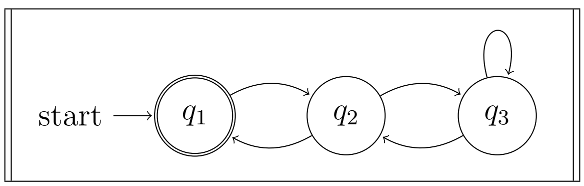


Table 4.6: $N(3, 5) = 1$ abstract NFA witnessing $A_N(x) \leq 3, |x| = 5$.

We can now classify all strings with complexity at most 3.

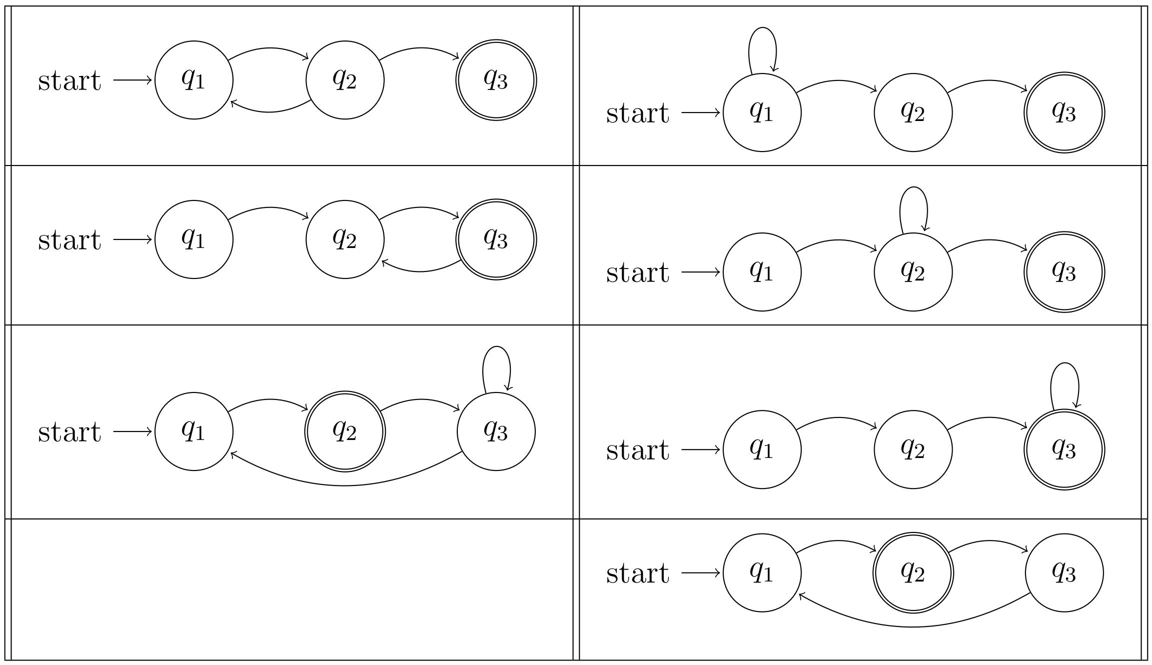


Table 4.7: $N(3,6) = 7$ abstract NFAs witnessing $A_N(x) \leq 3, |x| = 6$.

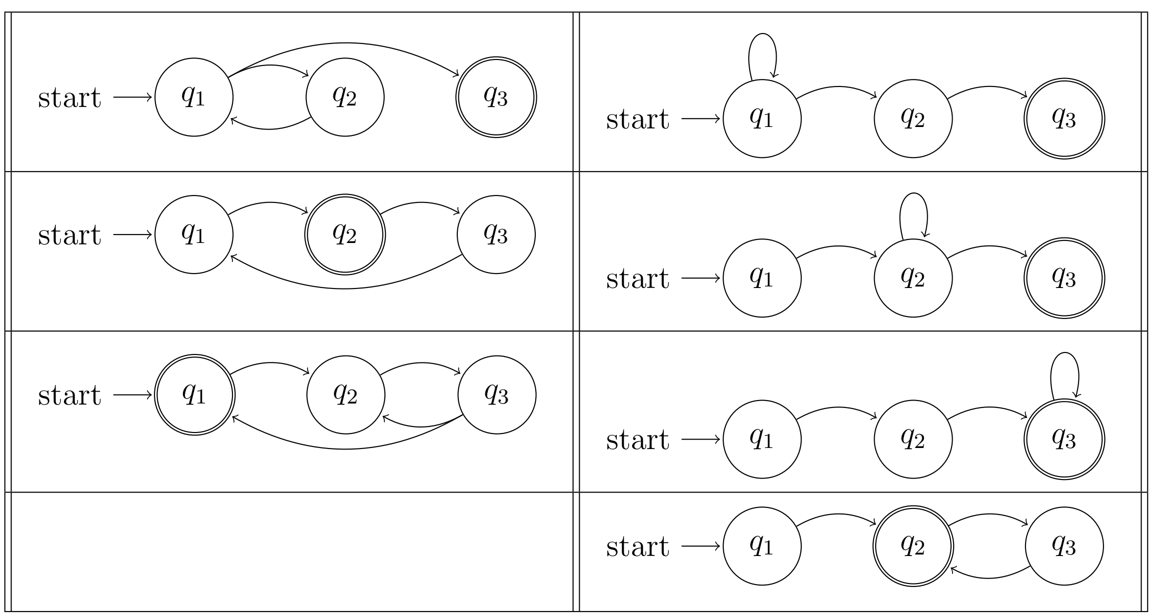


Table 4.8: $N(3,7) = 7$ abstract NFAs witnessing $A_N(x) \leq 3, |x| = 7$.

<u>If $A_N(x) = 3$, $|x| = 6$, then $x =$:</u>

- $(x_1)^4 x_2 x_3$
- $x_1(x_2)^4 x_3$
- $x_1 x_2 (x_3)^4$
- $(x_1 x_2)^2 x_1 x_3$
- $x_1 (x_2 x_3)^2 x_2$
- $x_1 x_2 (x_3)^2 x_4 x_1$
- $(x_1 x_2 x_3)^2$

<u>If $A_N(x) = 3$, $|x| = 7$, then $x =$:</u>

- $(x_1)^5 x_2 x_3$
- $x_1(x_2)^5 x_3$
- $x_1 x_2 (x_3)^5$
- $(x_1 x_2)^3 x_3$
- $x_1 (x_2 x_3)^3$
- $(x_1 x_2 x_3)^2 x_1$
- $x_1 (x_2 x_3)^2 x_2 x_4$

<u>If $A_N(x) = 3$, $|x| \geq 8, |x|$ even, then $x =$:</u>

- $(x_1)^m x_2 x_3$
- $x_1(x_2)^m x_3$
- $x_1 x_2 (x_3)^m$
- $(x_1 x_2)^m x_2 x_3$
- $x_1 (x_2 x_3)^m x_2$
- $|x|$=3m, $(x_1 x_2 x_3)^m$
- $|x|$=3m+1, $(x_1 x_2 x_3)^m x_2$
- $|x|$=3m+2, $(x_1 x_2 x_3)^m x_2 x_3$

<u>If $A_N(x) = 3$, $|x| \geq 9, |x|$ odd, then $x =$:</u>

- $(x_1)^m x_2 x_3$
- $x_1(x_2)^m x_3$
- $x_1 x_2 (x_3)^m$
- $(x_1 x_2)^m x_2$
- $x_1 (x_2 x_3)^m$
- $|x|$=3m, $(x_1 x_2 x_3)^m$
- $|x|$=3m+1, $(x_1 x_2 x_3)^m x_2$
- $|x|$=3m+2, $(x_1 x_2 x_3)^m x_2 x_3$

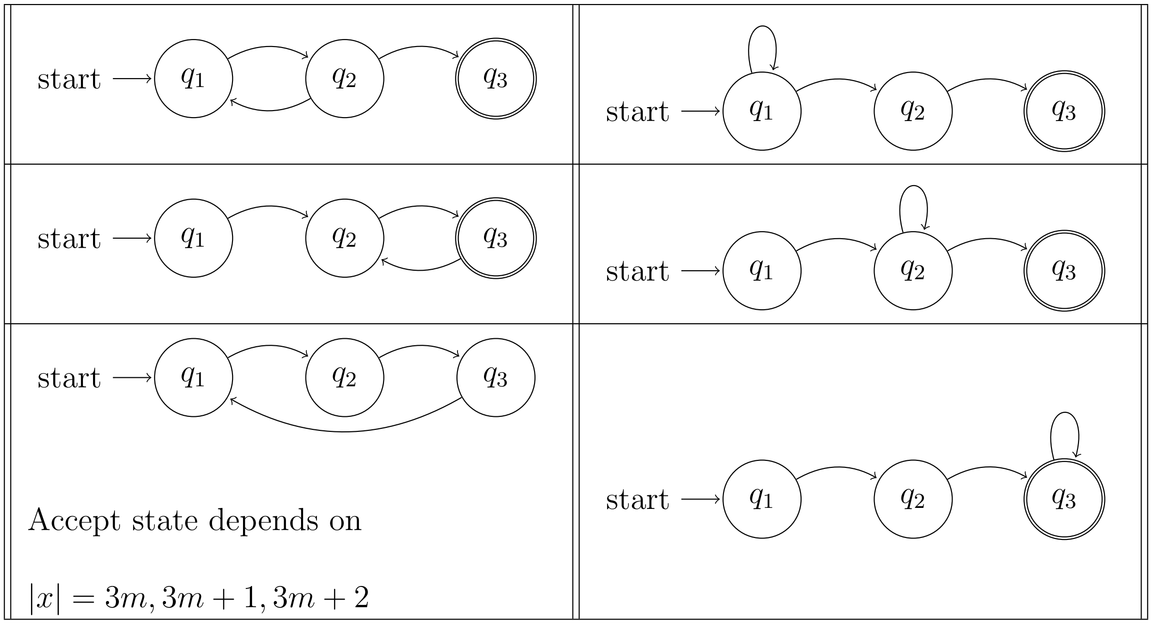


Table 4.9: $N(3, n) = 6$ abstract NFAs witnessing $A_N(x) \leq 3, |x| > 6, |x|$ even.

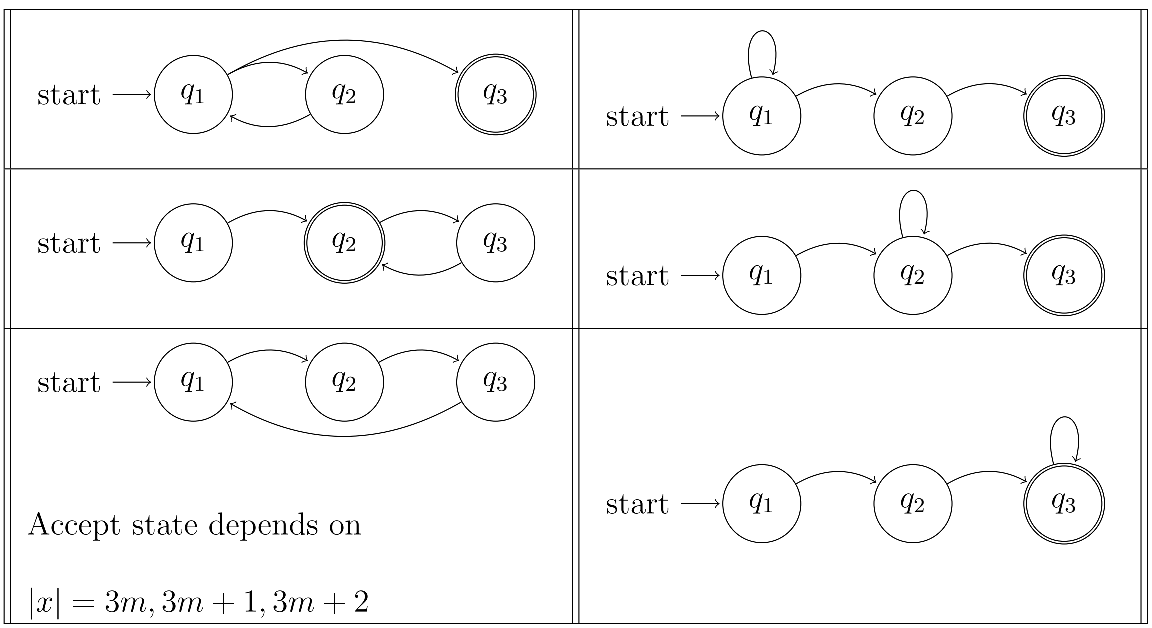


Table 4.10: $N(3, n) = 6$ abstract NFAs witnessing $A_N(x) \leq 3, |x| > 7, |x|$ odd.

Now with the addition of Table 4.11, and knowing $A_N(x) \leq 5$, we can find the complexity of all strings with $|x| = 8$.

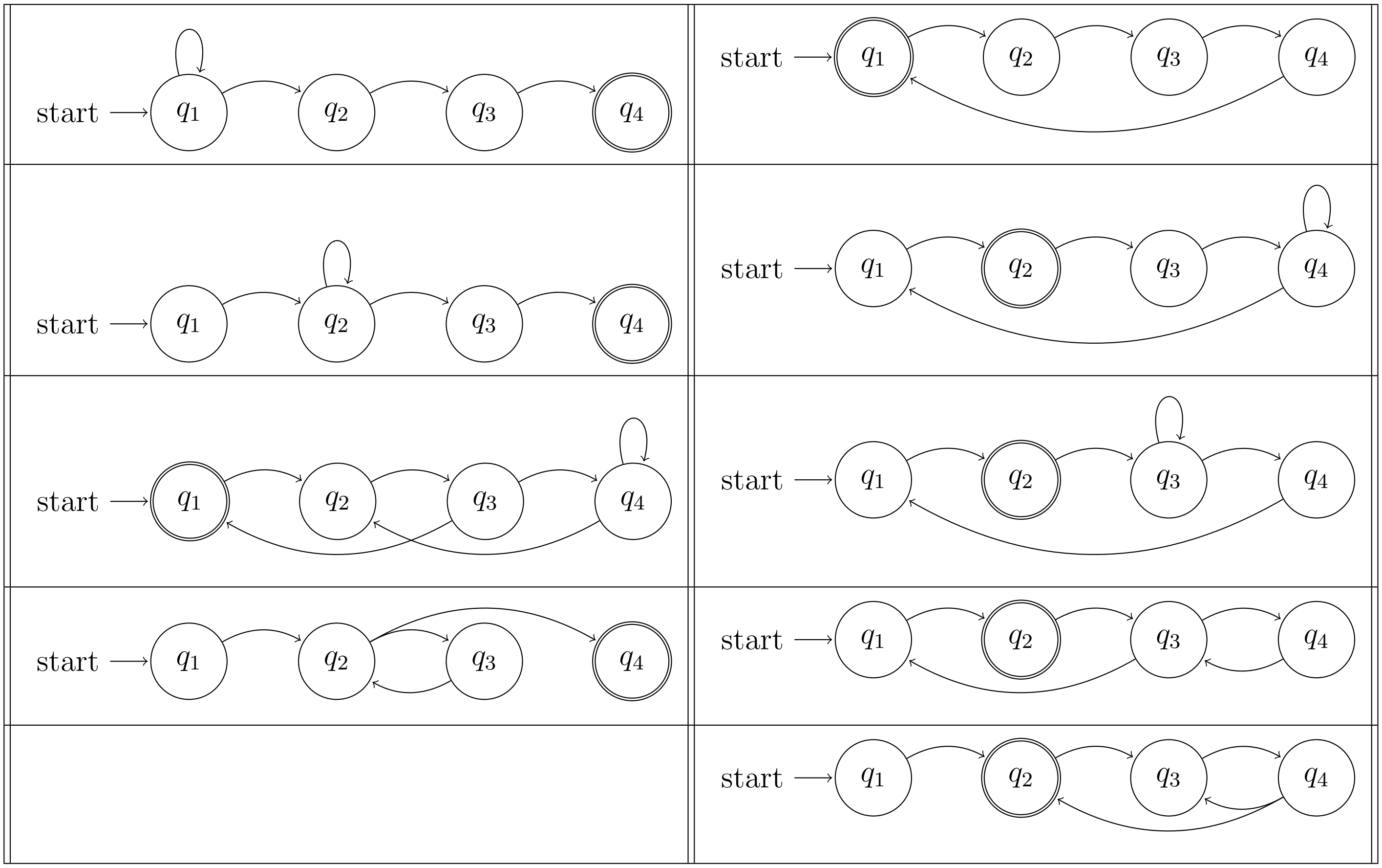


Table 4.11: $N(4,8) = 9$ abstract NFAs witnessing $A_N(x) \leq 4, |x| = 8$.

$\underline{A_N(x) \leq 4,\ |x| = 8:}$

- $(x_1)^5 x_2 x_3 x_4$
- $x_1 (x_2)^5 x_3 x_4$
- $(x_1 x_2 x_3 x_4)^2$
- $x_1 x_2 x_3 (x_4)^3 x_5 x_1$
- $x_1 x_2 (x_3)^3 x_4 x_5 x_1$
- $x_1 x_2 x_3 (x_4)^2 x_5 x_2 x_6$
- $x_1 (x_2 x_3)^3 x_4$
- $x_1 x_2 (x_3 x_4)^2 x_5 x_1$
- $x_1 x_2 (x_3 x_4)^2 x_3 x_5$

N(q,n)	**#**
N(1,1)	1
N(1,2)	1
N(1,3)	1
N(1,4)	1
N(1,5)	1
N(1,6)	1
N(1,7)	1
N(1,8)	1

N(q,n)	**#**
N(2,1)	x
N(2,2)	1
N(2,3)	1
N(2,4)	3
N(2,5)	3
N(2,6)	3
N(2,7)	3
N(2,8)	3

N(q,n)	**#**
N(3,1)	x
N(3,2)	x
N(3,3)	x
N(3,4)	1
N(3,5)	1
N(3,6)	7
N(3,7)	7
N(3,8)	6

N(q,n)	**#**
N(4,1)	x
N(4,2)	x
N(4,3)	x
N(4,4)	x
N(4,5)	x
N(4,6)	1
N(4,7)	1
N(4,8)	9

Table 4.12: $N(q,n)$ for $q \leq 4$, $n \leq 8$.

We now list the complexity of strings up to length eight which start with zero.

String	$A_N(x)$
0	1

String	$A_N(x)$
0	1
00	1
01	1

String	$A_N(x)$
000	1
001	2
010	2
011	2

String	$A_N(x)$
0000	1
0001	2
0010	3
0011	3
0011	3
0100	3
0101	2
0110	3
0111	2

Table 4.13: $A_N(x)$ for $|x| \leq 4$.

String	$A_N(x)$
00000	1
00001	2
00010	3
00011	3
00100	3
00101	3
00110	3
00111	3
01000	3
01001	3
01010	2
01011	3
01100	3
01101	3
01110	3
01111	2

String	$A_N(x)$
000000	1
000001	2
000010	3
000011	3
000100	4
000101	4
000110	4
000111	4
001000	4
001001	3
001010	3
001011	4
001100	3
001101	4
001110	3
001111	3
010000	3
010001	4
010010	3
010011	4
010100	3
010101	2
010110	4
010111	4
011000	4
011001	4
011010	4
011011	3
011100	3
011101	4
011110	3
011111	2

String	$A_N(x)$
0000000	1
0000001	2
0000010	3
0000011	3
0000100	4
0000101	4
0000110	4
0000111	4
0001000	4
0001001	4
0001010	4
0001011	4
0001100	4
0001101	4
0001110	4
0001111	4
0010000	4
0010001	4
0010010	3
0010011	4
0010100	3
0010101	3
0010110	4
0010111	4
0011000	4
0011001	4
0011010	4
0011011	4
0011100	4
0011101	4
0011110	4
0011111	3

String	$A_N(x)$
0100000	3
0100001	4
0100010	4
0100011	4
0100100	3
0100101	4
0100110	4
0100111	4
0101000	4
0101001	4
0101010	2
0101011	3
0101100	4
0101101	4
0101110	4
0101111	4
0110000	4
0110001	4
0110010	4
0110011	4
0110100	4
0110101	4
0110110	3
0110111	4
0110001	4
0111000	4
0111001	4
0111010	4
0111011	4
0111100	4
0111101	4
0111110	3
0111111	2

Table 4.14: $A_N(x)$ for $5 \leq |x| \leq 7$.

String	$A_N(x)$
0000000	1
00000001	2
00000010	3
00000011	3
00000100	4
00000101	4
00000110	4
00000111	4
00001000	5
00001001	5
00001010	5
00001011	5
00001100	5
00001101	5
00001110	5
00001111	5
00010000	5
00010001	4
00010010	5
00010011	5
00010100	4
00010101	4
00010110	4
00010111	5
00011000	4
00011001	4
00011010	5
00011011	5
00011100	4
00011101	4
00011110	4
00011111	4

String	$A_N(x)$
00100000	4
00100001	4
00100010	4
00100011	5
00100100	4
00100101	4
00100110	5
00100111	5
00101000	4
00101001	5
00101010	3
00101011	4
00101100	5
00101101	5
00101110	5
00101111	5
00110000	5
00110001	5
00110010	5
00110011	4
00110100	5
00110101	5
00110110	5
00110111	5
00111000	4
00111001	4
00111010	4
00111011	5
00111100	4
00111101	4
00111110	4
00111111	3

String	$A_N(x)$
01000000	3
01000001	4
01000010	4
01000011	4
01000100	4
01000101	5
01000110	4
01000111	4
01001000	5
01001001	5
01001010	5
01001011	5
01001100	5
01001101	5
01001110	5
01001111	5
01010000	5
01010001	5
01010010	5
01010011	5
01010100	3
01010101	2
01010110	4
01010111	5
01011000	5
01011001	5
01011010	4
01011011	4
01011100	4
01011101	5
01011110	4
01011111	4

String	$A_N(x)$
01100000	4
01100001	5
01100010	4
01100011	4
01100100	5
01100101	5
01100110	4
01100111	4
01101000	4
01101001	5
01101010	4
01101011	4
01101100	5
01101101	5
01101110	5
01101111	5
01110000	5
01110001	5
01110010	5
01110100	5
01110101	5
01110110	5
01110111	4
01111000	4
01111001	5
01111010	4
01111011	4
01111100	4
01111101	5
01111110	3
01111111	2

Table 4.15: $A_N(x)$ for $|x| = 8$.

REFERENCES

[1] I. Althofer. Tight lower bounds for the length of word chains. *Inform. Process. Lett.* **34** (1990), 275-276.

[2] J. Arndt. *Matters Computational: Ideas, Algorithms, Source Code.* Springer, 2011.

[3] E. Bach and J. Shallit. *Algorithmic Number Theory.* The MIT Press, 1996.

[4] B. Bunch. *Mathematical Fallacies and Paradoxes.* Dover Publications, Mineola, N.Y., 1997.

[5] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, Hoboken, NJ, 2006.

[6] A. A. Diwan. A new combinatorial complexity measure for languages. Technical report, Computer Science Group, Tata Institute, Bombay, 1986.

[7] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, 1985.

[8] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 1997.

[9] D. Poole. *Linear Algebra: A Modern Introduction.* Brooks/Cole, Boston MA, 2011.

[10] R. Sedgewick. *Algorithms in Java.* Pearson Education, Boston MA, 2003.

[11] J. Shallit and M. Wang. *Automatic Complexity of Strings.* Journal of Automata, Languages, and Combinatorics **vol.6** (2011), 537-554.

[12] M. Sipser. *Introduction to the Theory of Computation*. PSW Publishing, Boston MA, 1997.