ISSN 0304-9892 (Print) ISSN 2455-7463 (Online)
www.vijnanaparishadofindia.org/jnanabha
***Jñānābha*, Vol. 53(2) (2023), 236-241**
(Dedicated to Professor V. P. Saxena on His 80^{th} *Birth Anniversary Celebrations)*

ACCEPTABLE STRINGS IN AN AUTOMATON

Mridul Dutta[1] and Padma Bhushan Borah[2]
[1]Department of Mathematics, Dudhnoi College,Goalpara, Assam, India-783124
[2]Department of Mathematics, Cotton University, Assam, India-781001
Email: mridulduttamc@gmail.com, padmabhushanborah@gmail.com

(Received: September 30*,* 2022*; In format: October* 08*,* 2022*; Revised: August* 15*,* 2023*;*
Accepted: November 26*,* 2023*)*

DOI: https://doi.org/10.58250/jnanabha.2023.53228

Abstract

This paper is a presentation and discussion of two proofs of a theorem. The theorem is a statement about a specific type of sequence of inputs where all multiples of fixed denominations are accepted as inputs of an automaton. The theorem has interesting implications for accepted strings in a finite automaton in general setting. Here, we have examined different methods for proving the theorem. We present one analytic proof by using automata theory with graph theoretic concepts and another proof by utilizing the ordered partition of number theory. We also illustrate the results with the help of examples.
2020 Mathematical Sciences Classification: 03D05, 11B85, 68R15.
Keywords and Phrases: Finite Automata, Acceptable Strings, Ordered Partition.

1 Introduction

An automaton (in plural Automata) is an abstract self-operating machine which follows a predetermined sequence of operation automatically gives an output from input. Here input may be energy, information, materials, etc. The system works without the intervention of man. Automata theory plays a major role in huge applied areas. The most significant areas include communication, transportation, health care, electronic banking, etc. Mainly finite automata are significant in many different areas, including Electrical Engineering, Linguistics, Computer Science, Philosophy, Biology, Mathematics, etc. In Computer science, automata widely used in text processing, Compilers, Software and hardware design, network protocol, etc. [6]. Many authors have done their work on the string of automaton for a long time. Yu et al. [5] presented symbolic string verification: An automata-based approach. Aydin et al. [1] had done their work on Automata-based model counting for string constraints. Most recently, Yue et al. [8] developed the language acceptability of finite automata based on theory of semi-tensor product of matrices. Dobronravov et al. [3] introduced the length of the shortest strings accepted by two-way finite automata. As a result of the techniques used in the aforementioned works, we are continuing our research on acceptable strings in an automaton. The originality of the paper is that we primarily provide many proofs utilizing entirely distinct methodologies.

2 Preliminaries

A *finite state automaton* consists of a finite set of states and a set of transitions from state to state that occurs on input symbols from a set of alphabets. An *alphabet* is a finite, non-empty set of symbols denoted by A, e.g. $A = \{0, 1\}$, the set of binary alphabet. A *string* (or word) is a finite sequence of symbols chosen from the set A, e.g. 01101, 01, 1, 0 are some strings over the set alphabet $A = \{0, 1\}$. A *Deterministic Finite Automata* can be formally defined as a 5-tuple $\sum = (Q, A, \delta, q_0^*, F)$ where $Q\,(\neq \phi)$ is a finite set of states, A is a finite non-empty set of inputs, $\delta : Q \times A \to Q$ is defined by $\delta\,(q_0^*, a) = q_1; q_0^*, q_1 \in Q, a \in A, q_0^*$, is the initial state, F is the set of final states and $F \subseteq Q$. A string x is *accepted* by finite state automata $\sum = (Q, A, \delta, q_0^*, F)$ if $\delta\,(q_0^*, x) = p$ for some $p \in F$. A final state is also called an *accepting state*. The initial state is denoted by an arrow mark and the final state is denoted by a double circle. The input is accepted when all input is read and match by transitions and the automaton is in a final state [6].

A finite-state automaton is a machine that constructs computing by reading a one-way read-only tape. The input is produced up of words written on the tape. The written words use a describe alphabet which is called the input alphabet and the words create a string. The Finite automata will be produced up of the input-output relations at every state and also the modifications of the states that will appear in receiving the input at a particular state. At the end of the process, it becomes visible whether the input is accepted

or rejected by the automaton machine. Also, *Deterministic* refers to the distinctiveness of the computation. The finite automata are called deterministic finite automata if the machine reads an input string one symbol at a time [6, 8]. *Tree automata* are state machines. They deal with tree structures rather than the strings of the more traditional state machine [4].

In Combinatorics and Number theory, a partition of a positive integer n, also called an integer partition, is a way of writing n as a sum of natural numbers. If order matters, the sum becomes a composition or ordered partition. Thus, a composition or an *ordered partition* of an integer n is a way of writing n as the sum of a sequence of positive integers [2].

3 Main Results

In this section, we present our result with different proofs. Also we discuss some examples.

Theorem 3.1. *In a finite automata, if q_0 is the initial state, q_m is the final state with $m = nk$, where $m, n, k \in \mathbb{N}$, and automata accept inputs of denomination ln, where $1 \leq l \leq k$ then the number of acceptable strings in the automata is $2^{(k-1)}$.*

Proof. Let $l_s n, s \in \{1, 2, ..., r\}, 1 \leq r \leq k$ is a sequence of automaton acceptable inputs. Then, we have $nk = (l_1 + l_2 + ... + l_r)n$. i.e. $k = l_1 + l_2 + ... + l_r$. So, without loss of generality, let $n = 1$ be the lowest possible denomination accepted by the automaton. Hence $m = k$. Now, Consider the finite tree automata with states $\{q_0, q_1, ...q_k\}$ with a direct transition function from state q_i to state q_j if and only if $i < j$. q_l has stored value $l, 0 \leq l \leq k$. We get the final state when we reach the value $m = k$, i.e., when we reach state q_k. Now, going from state q_i to state $q_j, i < j$, takes us from value i to value j; i.e. input $(j - i)$ is added to the present value i.

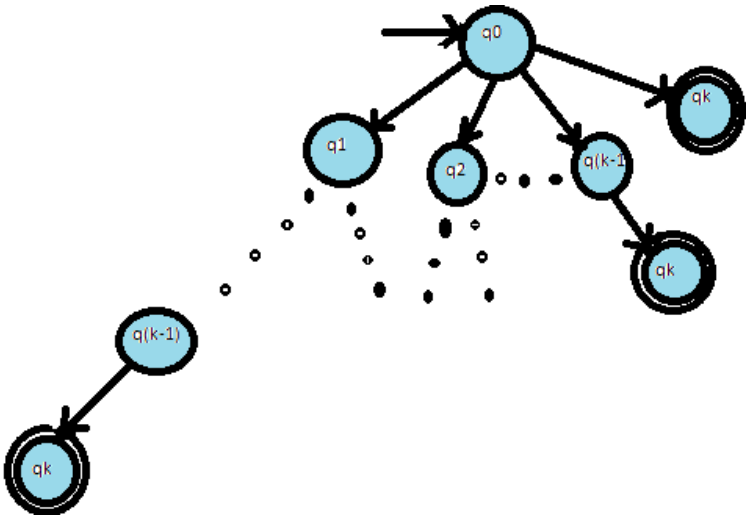

***Figure* 3.1**. A state diagram of a finite Tree automata

A path (transition sequence) from state q_0 to state q_k will be define an automaton acceptable strings. Eg. When $k = 3$ in the tree automaton below, we have $(q_0, q_1, q_2, q_3), (q_0, q_1, q_3), (q_0, q_3), (q_0, q_2, q_3)$ are acceptable strings.

Since q_0 is always the starting states and q_k the final states, we have each subset of $S = \{q_1, q_2, ..., q_{k-1}\}$ correspond to a unique (path) transition function from q_0 to q_k and vice-versa. Therefore, there are a total of $2^{|S|} = 2^{k-1}$ transition functions or paths from q_0 to q_k and therefore, there is 2^{k-1} acceptable strings of the finite automaton. □

Alternate proof. If $l_s n, s \in \{1, 2, ..., r\}, 1 \leq r \leq k$ is an accepted sequence of inputs, then, we have $nk = (l_1 + l_2 + ... + l_r)n$. i.e. $k = l_1 + l_2 + ... + l_r$.
So, without loss of generality, let $n = 1$ be the lowest possible denomination accepted by the automaton. Hence we get $m = k$. Now $k = l_1 + l_2 + ... + l_r$. means $\{l_1, l_2, ..., l_r\}$ forms a partition of k.

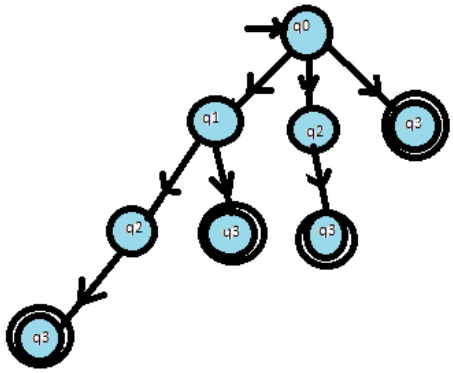

***Figure* 3.2**. A state diagram of a Tree automata with 3 states

Let N = No. of acceptable strings of the automaton.
We Claim that $N = 2^{k-1}$.

Clearly, any acceptable strings, or sequence of inputs form an ordered partition of m and vice-versa.Therefore, there are as many acceptable strings as there are ordered partition of m. Now $m = 1+1+...+1(k\text{-times of } 1) = k$, there are $(k-1)$'$+'$ signs between the '$k'1s$. We define an operationdeleting a + sign as replacing the '$1's$ joined by the + signs to be deleted with their sum, keeping the remaining + signs undisturbed. With this interpretation, each set of choice for '$+'$ signs to be deleted corresponds to a unique ordered partition of 'm and vice-versa. Eg. $m = 3, 3 = 1+1+1$, Choosing the 2nd '$+'$ sign correspond to the ordered partition $1+(1+1)$ i.e. $1+2$,and the ordered partition $2+1$ i.e. $(1+1)+1$ corresponds to choosing and deleting the 1st '$+'$ sign. Similarly choosing both the '$+'$ sign correspond to $(1+1+1) = 3$ and not choosing any of the + sign corresponds to the partition $1+1+1$ of 3 etc. Since there are 2 choices for each + sign, viz. to delete or not to delete, and there are a total of $(k-1)+$ signs, so there are 2^{k-1} such choices in total and as such there are 2^{k-1} ordered partition of m. Hence, $N = 2^{k-1}$.

Example 3.1. In a finite automata, if q_0 is the initial states, q_m is the final state with $n = 1, m = k = 4$, Allowed denomination i.e inputs are $S = 1, 2, 3, 4$. There are $2^{4-1} = 8$ accepted sequences of inputs by the

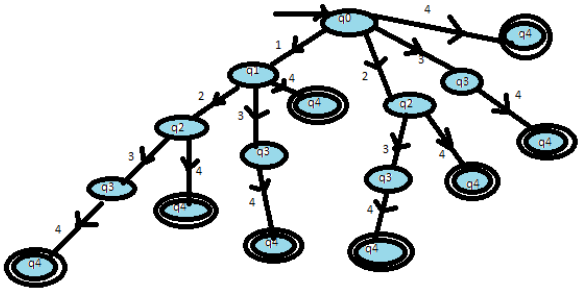

***Figure* 3.3**. A state diagram of a Tree automata with 4 states

finite automata. They are (1,2,3,4), (1,2,4), (1,3,4), (2,3,4), (2,4), (3,4), (1,4), and (4). These are obtained by traversing all the transition functions from the state q_0 to state q_4.

Alternatively, For $4 = 1+1+1+1$, there are four '1s and 3'$+$ signs. We define a function f_3 on all binary strings of length 3 to the ordered partition of 4. Now,
$f_3(000) = 1+1+1+1$,
$f_3(001) = 1+1+(1+1) = 1+1+2$,
$f_3(010) = 1+(1+1)+1 = 1+2+1$,
$f_3(011) = 1+(1+1+1) = 1+3$,
$f_3(100) = (1+1)+1+1 = 2+1+1$,

$f_3(101) = (1+1)+(1+1) = 2+2$,
$f_3(110) = (1+1+1)+1 = 3+1$,
$f_3(111) = (1+1+1+1) = 4$.
Corresponding to each binary string of length 3 we get an ordered partition of 4 which in turn corresponds to an acceptable sequence of inputs viz. $(1,2,3,4), (1,2,4), (1,3,4), (1,4), (2,3,4), (2,4), (3,4), (4)$ respectively and these are precisely 8 in numbers.

Example 3.2. In a finite automata, if q_0 is the initial states, q_m is the final state with $n = 5, m = 5k = 20$, Allowed denomination are multiple of 5 i.e inputs are $S = 5, 10, 15, 20$. There are $2^{4-1} = 8$ accepted

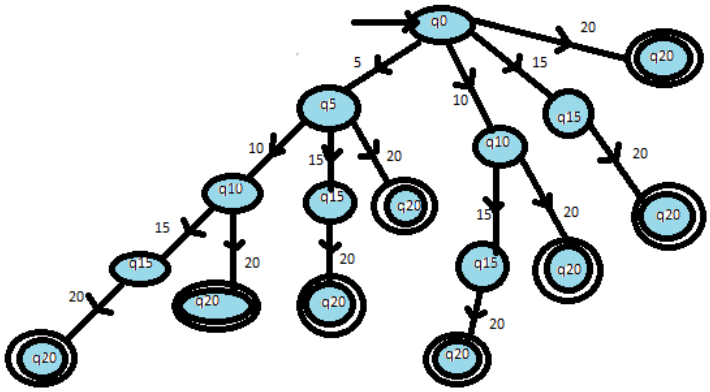

***Figure* 3.4**. A state diagram of a Tree automata with 4 states

sequences of inputs by the finite automata. They are (5,10,15,20), (5,10,20), (5,15,20), (5,20), (10,15,20), (10,20), (15,20), and (20). These are obtained by traversing all the transition functions from the state q_0 to state q_{20}.

Alternately, For $20 = 5+5+5+5$, there are four 5s and $3+$ signs. We define a function g_3 on all binary strings of length 3 to the ordered partition of 20.
Now,
$g_3(000) = 5+5+5+5$,
$g_3(001) = 5+5+(5+5) = 5+5+10$,
$g_3(010) = 5+(5+5)+5 = 5+10+5$,
$g_3(011) = 5+(5+5+5) = 5+15$,
$g_3(100) = (5+5)+5+5 = 10+5+5$,
$g_3(101) = (5+5)+(5+5) = 10+10$,
$g_3(110) = (5+5+5)+5 = 15+5$,
$g_3(111) = (5+5+5+5) = 20$.
Corresponding to each binary string of length 3 we get an ordered partition of 20 which in turn corresponds to an acceptable sequence of inputs (5,10,15,20), (5,10,20), (5,15,20), (5,20), (10,15,20), (10,20), (15,20), and (20) respectively and these are precisely 8 in numbers.
We note that in both the examples there are exactly 8 accepted sequence of inputs. This was bound to happen since we have $5k = 5l_1 + 5l_2 + \ldots + 5l_r \Leftrightarrow k = l_1 + l_2 + \ldots + l_r$.

We conclude with a final example.

Example 3.3. In a finite automata, if q_0 is the initial states, q_m is the final state with $n = 1, m = k = 5$, Allowed denomination i.e inputs are $S = 1, 2, 3, 4, 5$ There are $2^{5-1} = 16$ accepted sequences of inputs by the finite automata. They are (1,2,3,4,5), (1,2,3,5), (1,2,4,5), (1,3,4,5), (2,3,4,5), (1,2,5), (1,4,5), (3,4,5), (1,5), (4,5), (2,5), (3,5), (2,4,5), (1,3,5), (2,3,5),and (5). These are obtained by using transition functions from the state q_0 to state q_5.

Alternatively, For $5 = 1+1+1+1+1$, there are five 1s and $4+$ signs. We define a function h_4 on all binary strings of length 4 to the ordered partition of 5. Now,

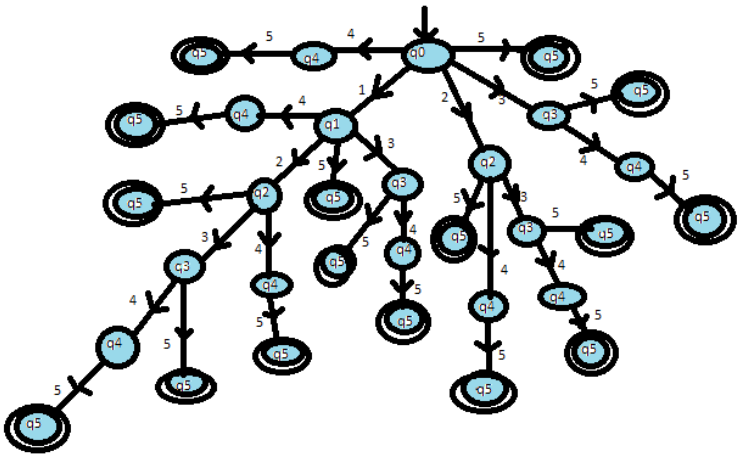

***Figure* 3.5**. A state diagram of a Tree automata with 5 states

$h_4(0000) = 1+1+1+1+1$,
$h_4(0001) = 1+1+1+(1+1) = 1+1+1+2$,
$h_4(0010) = 1+1+(1+1)+1 = 1+1+2+1$,
$h_4(0011) = 1+(1+1)+1+1 = 1+2+1+1$,
$h_4(0100) = (1+1)+1+1+1 = 2+1+1+1$,
$h_4(0101) = 1+1+(1+1+1) = 1+1+3$,
$h_4(0110) = 1+(1+1+1)+1 = 1+3+1$,
$h_4(0111) = (1+1+1)+1+1 = 3+1+1$,
$h_4(1000) = 1+(1+1+1+1) = 1+4$,
$h_4(1001) = (1+1+1+1)+1 = 4+1$,
$h_4(1010) = (1+1)+(1+1+1) = 2+3$,
$h_4(1011) = (1+1+1)+(1+1) = 3+2$,
$h_4(1100) = (1+1)+(1+1)+1 = 2+2+1$,
$h_4(1101) = 1+(1+1)+(1+1) = 1+2+2$,
$h_4(1110) = (1+1)+1+(1+1) = 2+1+2$,
$h_4(1111) = (1+1+1+1+1) = 5$.
Corresponding to each binary string of length 4 we get an ordered partition of 5 which in turn corresponds to an acceptable sequence of inputs viz. (1,2,3,4,5), (1,2,3,5), (1,2,4,5), (1,3,4,5), (2,3,4,5), (1,2,5), (1,4,5), (3,4,5), (1,5), (4,5), (2,5), (3,5), (2,4,5), (1,3,5), (2,3,5), and (5) respectively and these are precisely 16 in numbers.

4 Conclusion

We have presented two different proofs of a theorem in automata, each being different in its approach. We looked at an Graph theoretic proof using the concept of a tree, and our second proof used another very interesting concept of ordered partition or composition of numbers from Combinatorics and Number theory. Finally, we examplified the theorem with both the approaches.

References

[1] A. Aydin, L. Bang and T. Bultan, Automata-based model counting for string constraints, *In International Conference on Computer Aided Verification*, Cham: Springer International Publishing, (2015), 255-272.
[2] D. M. Burton, *Elementary number theory*, McGraw-Hill Education, New York, USA, 2007.
[3] E. Dobronravov, N. Dobronravov and A. Okhotin, On the length of shortest strings accepted by two-way finite automata, *Fundamenta Informaticae*, **180**(4) (2021), 315-331.
[4] F. Harary, *Graph Theory*, Narosa Publishing House, India, Reprint, 2013.
[5] F. Yu, T. Bultan, M. Cova and O. H. Ibarra, Symbolic string verification: An automata-based approach, *International SPIN Workshop on Model Checking of Software*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[6] J. E. Hopcroft, R. Motwani and V. Ullman, *Introduction to Automata Theory, Language and Computations*, Addison Wesley, USA, 2001.
[7] K. L. P. Mishra and M. Chandrasekarn, *Theory of computer science; Automata, Languages and computation*, Third edition, Prentice-Hall of India, 2007.
[8] J. Yue, Y. Yan and Z. Chen, Language acceptability of finite automata based on theory of semi-tensor product of matrices, *Asian journal of control*, **21**(6) (2019), 2634-2643.