



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5

Issue: V

Month of publication: May 2017

DOI:

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Design of Deterministic Finite Automata using Pattern Matching Strategy

V. N. V Srinivasa Rao¹, Dr. M. S. S. Sai²

Assistant Professor, ² Professor, Department of Computer Science and Engineering
KKR & KSR Institute of Technology and Sciences, Guntur, A.P, India.

Abstract: *An automaton is an entity work for specific computation and designed for that computation. Based on the problem automaton can be designed to satisfy the given problem. Automata have finite and infinite types. Finite automata contain again two types Deterministic Finite Automata and Non-Deterministic Finite automata. In this paper, the proposed strategy is the design of Deterministic Finite Automata using Pattern matching strategy. Based on the given input string or problem it divides the given string as patterns and matches those patterns to states and transitions for making entire automaton. This strategy has different from the general procedure.*

Keywords: *Deterministic Finite Automata, Pattern Matching Strategy, Automaton.*

I. INTRODUCTION

It shows an implementation of designing of Deterministic finite automata in an efficient manner. Normally computation is a concept common to all machines which involved in computing. In order to understand fully regarding machines power and drawbacks. First, it needs to know the computational models and computers.

A machine or an automaton is a model/procedure to solve a particular problem. Before developing a machine first it needs to know the problem clearly and how it can be solved, and methodology to solve that problem. Based on that methodology it will design a machine or automaton in the correct way. And it can be able to judge whether a problem is solvable or not? And also machine accepts which type of problems it needs to know.

As a computer engineer/scientist, these two points must be able to answer while designing a machine for a problem. When we study computability, we study problems in an isolated sense. For example

- A. The addition is the problem of returning a third number that is the sum of two given numbers.
 - B. Traveling salesmen problem is one in which a list of distances between some number of cities are given and the person is asked to find the shortest route so that he visits each city once and returns to the start
 - C. The halting problem is one in which a program is given some appropriate input and it needs to be decided whether the program, when running on that input, loops forever or halts.
- 1) *Problem B and C are the Two Problems of Computer Interest:* In both these problems, the statement of the problem does not give the actual values needed to provide the result but just tells what kind of objects they are. A set of actual values for a problem is called an instance of the problem. In all these problems an instance is required, that is the input and the relationship between the input and the output is to be understood. In order to solve these problems, there are certain things one should know. Can it be solved algorithmically- is there any definite procedure that solves any instance of the problem in a finite amount of time? How hard is it to solve? What kind of resources are needed and how much of these resources are required? To find a solution to a problem algorithmically, it is required to know whether an algorithm for solving it exists.

II. LITERATURE SURVEY

Pattern matching is a major process based on signature and it is used to intrusion detection and intrusion prevention system. To identify the security threats by matching the signature patterns. And the pattern matching is relevant to reduce the memory space requirement of a Deterministic Finite Automata. The size of DFA can be reduced by some techniques like transition condensing. The transition condensing technique trims reduces the size of Deterministic Finite Automata. In Character set condensing, alphabet set of signature patterns trims down the number of bits represent to every state in DFA[1]

Pattern matching scheme described in this paper is well suited for applications used for the searching process which is based on text strings and it supports a large number of keywords. Some information retrieval systems compute and index for a text file to allow searches to conducted without having to scan all of the text string. [2]

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Storage space of Deterministic finite automata is the very important concern and a large amount of memory is essential to store transition function and this paper reduces the size of Deterministic finite automata and it is in the form of regular expressions. And the evaluation of performance is done [3]

New compressed technique for Deterministic finite automata for Delta finite automata, it reduces the number of states and transitions and it is based on adjacent and common transitions and also it is convenient to store any differences between them. [4]

In hybrid automaton which addresses this issue by combining the benefits of deterministic and non-deterministic finite automata. This can be tested on smart rule sets and we validate it on real traffic traces. It is comparative analysis for the worst case behavior of this technique and to traditional ones. [5]

Several mechanisms have proposed to enhance the performance of regular expressions parsers which are orthogonal with respect to each other and hence they can employ in unison. And these mechanisms can reduce the memory requirements of today's state-of-the-art on networking systems and links, these mechanisms can improve the protection and throughput on various signatures. [6]

Automata-based representations and related algorithms have been applied to address several problems in information security, and introduced the extended finite automata (XFAs), primary intention of XFAs is to match the signature in Network Intrusion Detection Systems (NIDS) [7]

A new representation for deterministic finite automata as Delta Finite Automata which considerably reduces the size of DFA in terms of states and transitions and requires a transition per input symbols and allows the fast matching. [8]

Exploring the multiple compression tables as a lightweight method for reducing the memory requirements of DFAs. It can evaluate this method on signature sets used in Cisco IPS and snort. Compared uncompressed DFAs, multiple ACTs achieve memory savings between a factor 4 and a factor of 70 at the cost of an increase in run time. [9]

It significantly reduces the amount of memory required, and addressing the challenges the conventional assumption that graph data structures must store pointers of bits to identify the successor nodes [10]

Pattern matching is a basic method used in security applications and it is widely used in INS Intrusion Detection System. It proposes the reduction techniques on transitions, state reduction, and character set reduction and bit reduction. [11]

Based on a formal definition a finite automaton is a collection of five objects: set of states, input alphabet. Rules for moving the start and accept states. Thus a mathematical model definition of a finite automaton is a five-tuple consisting of five objects.

A. Elements of Deterministic Finite Automata

$M = (Q, \Sigma, \delta, q_0, F)$

Q - Set of States

Σ - Input Symbols

δ - Transition Function

q_0 - Initial State

F - Final State/Accepted State

B. Operations of Deterministic Finite Automata

Initially, DFA is started from the initial state by receiving an input symbol from sequential tape and move to next state. This process moves on until end of the string and at the end of the string machine's state is the final state or accepting the state of a machine. Each move consumes one input symbol. When the end of the string is reached, the string is accepted if DFA is in one of the final states, else rejected. The working of DFA is shown in below figure 1:

Ordered Quinn-Tuple Specification of DFA

Formally, a DFA, M is a five-tuple

Formally, a DFA, M is a five tuple

$M = (Q, \Sigma, \delta, q_0, F)$

Q - Set of states $q_0, q_1, q_2, \dots, q_n$

Σ - input Symbols (characters, numbers and symbols)

δ - Transition function $\delta \times \Sigma \rightarrow Q$

q_0 - initial state $q_0 \in Q$

F - Final state/accepted state F subset of Q.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

C. Description of DFA

The transition function can be represented by a table or a diagram. In that, every transition has contained a state with an input symbol moves to next state. Showing every step as a transition [8].

D. Design of DFA

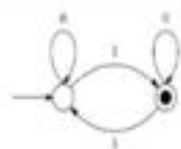
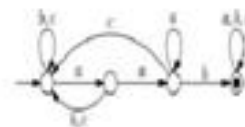


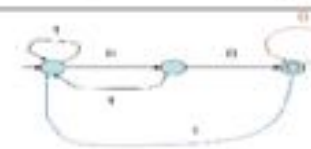
The basic design strategy for DFA is as follows:

- 1) Understand the condition for designing a language or grammar for designing a language.
- 2) Specify the start state for a diagram Find the initial, accepting and dead states in a finite automaton.
- 3) Check for every state for every character in input symbol can do a transition or not.
- 4) Design the transition table and diagram for the given language and DFA.
- 5) Test the DFA is valid for every string of the language is satisfied or not.

E. An Example of DFA:

Construct a DFA which accepts a language $L = \{w/w \text{ contains even no. of a's and even no. of b's}\}$ Here the condition that input contains an even no of series with an input symbol $\{a, b\}$. At the initial step, we need to find the language for the given condition. And then design a Deterministic Finite Automata for the following condition[8].

For Deterministic finite automata the language specifies below:

1. Give a DFA for $\Sigma = \{0, 1\}$ and $L = \{\text{strings that have an odd number of 1's and any number of 0's}\}$.	
2. Give a DFA for $\Sigma = \{a, b, c\}$ that $L = \{\text{accepts any string with aab as a substring}\}$.	
3. Give a DFA for $\Sigma = \{a, b\}$ that $L = \{\text{accepts any string with aababb as a substring}\}$.	
4. Give a DFA for the given Regular Expression $R = (0+1)^*00(0+1)^*$	
5. Give a DFA for the given Regular Expression $R = (0+1)^*00$	

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

III. PATTERN MATCHING STRATEGY DESIGN

This pattern matching strategy for the design of design is taken the input string and generates the patterns based on the power set of that string from null string to complete string and match the string as states accepting and if the state is matched to the pattern then the state is finalized for the design of an automaton [1].

For example: For a language condition to accept the strings which end with 111 then we will take for each of the characters from zero no. of characters to n no. of characters to combined as the string.

For a condition which ends with {111} the following string, patterns are: { ϵ , 1, 11, 111}, for every string specified by the language is accepted by some of the states and that states also part of the state transition diagram. For every state, there is a transition for every input symbol. And every time we have to check the rule of Deterministic Finite Automata [8].

A. The Rules for Designing a DFA are:

- 1) Firstly the states specified in deterministic finite automata are finite and not infinite states.
- 2) Next, the transitions for every state have at most the no of transitions related to input symbols. If the input symbols are n then the transitions for every state is at least n. For every input symbol, there must be a transition to every state [3].

Without satisfying these two strategies the DFA is not valid.

B. Sequential DFA Run

The sequential run of DFA uses the fact that there is always just one state active during the computation [7].

C. Algorithm 1(Basic Sequential Run of DFA)

- 1) *Input:* A transition table δ and a set F of final states of DFA, input text $T=t_1t_2 \dots t_n$, q is an active state, q_0 is Initial state
- 2) *Output:* Information whether DFA accepts whole input text or not
- 3) *Method:*

Step 1: $J \leftarrow 0$

Step 2: $Q \leftarrow q_0$

Step 3: While ($j \leq n$) do

Step 4: $q \leftarrow \delta$

Step 5: [q, t_j]

Step 6: $J \leftarrow j + 1$

Step 7: Endwhile

Step 8: if ($q \in F$) then

Step 9: write („The automaton accepts the input string.“)

Step 10: else

Step 11: write („The automaton does not accept the input string.“)

Step 12: end if

Note that the above algorithm of accepting automation can be useful only when purposes needed to know if the entire input text is accepted or not. Sometimes information about all reached final states is necessary [8]. In those cases, the modification is solving in the situation where the state and position of the last read symbol are taken out and this type of finite automaton used to solve but is not efficient when entire strings are not in the form. For those cases the below pattern matching algorithm is useful [7].

D. Algorithm 2 (Basic Sequential Run of DFA—Pattern Matching Version)

- 1) *Input:* A transition table δ and a set F of final states of DFA, input text $T=t_1, t_2 \dots t_n$, q is the active state, q_0 is an initial state
- 2) *Output:* Information about all reached final states
- 3) *Method:*

Step 1: $J \leftarrow 0$

Step 2: $q \leftarrow q_0$

Step 3: while ($j \leq n$) do Step 4: $q \leftarrow \delta[q, t_j]$

Step 5: $j \leftarrow j + 1$ Step 6: endwhile

Step 6: if ($q \in F$) then

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Step 7: write („Final state „q „reached position j.“)

Step 8: end if

Design a Deterministic Finite Automata with satisfies the strings ending with “100”.

Initial step is:

Step 1: Design the language for the given condition.

$L = \{100\}$

Step: 2 Design the Mathematical Representation for DFA , $M=(Q,\Sigma,\delta,q_0,F)$

Q/Σ	0	1
------------	---	---

q_0	-----	-----
-------	-------	-------

q_1

q_2

q_3

Q-

Strings end with ϵ accept by q_0 ---- ϵ

Strings end with 1 accept by q_1 -----1

Strings end with 10 accept by q_2 ----10

Strings end with 100 accept by q_3 ----100

In the first step to fill the q_0 transitions we check $\epsilon 0$ transition and $\epsilon 1$ transition. First one is $\epsilon 0$ is here above transition ends with any $\epsilon 0$ –no then cut ϵ and check only 0 transitions. There is no 0 transition also so it is gone with self transition. So $\delta: q_0 X 0 \rightarrow q_0$

For second transition is $\epsilon 1$ i.e we have to check is there any available string for $\epsilon 1$ in the table. Here in table there is no $\epsilon 1$ transition and cut the ϵ and check for the 1 transition in th given pattern, yes the 1 pattern is available in the string state is q_1 . So $\delta: q_0 X 1 \rightarrow q_1$

Q/Σ	0	1
q_0	q_0	q_1 ---- ϵ
q_1	-----	-----1
q_2	-----	-----10
q_3	-----	-----100

In the similar way we have to solve for the every transitions for every state.

Next state is q_1 , for q_1 we have to find the 0th transition and 1 transition. Or we have to find the transition for q_1 state with input symbol $\Sigma = \{0,1\}$

In the first step to fill the q_1 transitions we check 10 transitions and 11 transitions. First one is 10 is here above transition ends with q_2 . So $\delta: q_1 X 0 \rightarrow q_2$

For second transition is 11 i.e we have to check is there any available string for 11 in the table. Here in table there is no 11 transition and cut the 1 and check for the 1 transition in the given pattern, yes the 1 pattern is available in the string state is q_1 . So $\delta: q_1 X 1 \rightarrow q_1$

Q/Σ	0	1
q_0	q_0	q_1 ---- ϵ
q_1	q_2	q_1 -----1
q_2	-----	-----10
q_3	-----	-----100

Next state is q_2 , for q_2 we have to find the 0th transition and 1 transition. Or we have to find the transition for q_2 state with input symbol $\Sigma = \{0,1\}$

In the first step to fill the q_2 transitions we check 100 transition and 101 transitions. First one is 100 is here above transition ends with q_3 . So $\delta: q_2 X 0 \rightarrow q_3$

For second transition is 101 i.e we have to check is there any available string for 101 in the table. Here in table there is no 101 transition and cut the first 1 and check for the 01 transition in the given pattern, no there is no 01 string pattern available, then cut down the first 0, and check for only 1, yes the 1 pattern is available in the string state is q_1 . So $\delta: q_2 X 1 \rightarrow q_1$

Q/Σ	0	1
q_0	q_0	q_1 ---- ϵ

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

q1	q2	q1	-----1
q2	q3	q1	-----10
q3	-----	-----	----100

Next state is q3, for q3 we have to find the 0th transition and 1 transition. Or we have to find the transition for q3 state with input symbol $\Sigma = \{0,1\}$

In the first step to fill the q3 transitions we check 1000 transition and 1001 transitions. There is no string pattern of 1000, and we cut down the first 1 in pattern and check 000 string pattern in the sequence. And there is no 000 string pattern, again we cut down the first 0 from the string pattern. And again check for 00 string pattern, again 00 string pattern also not there, again we cut down first zero in the pattern and check the pattern available or not. Again 0 patterns also not available, we cut down that remaining zero also, now only null string is there i.e ϵ . So ϵ is available in pattern i.e q0. First one is 1000 is here above transition ends with q0. So $\delta: q3 \times 0 \rightarrow q0$

For second transition is 1001 i.e we have to check is there any available string for 1001 in the table. Here in table there is no 1001 transition and cut the first 1 and check for the 001 transition in the given pattern, no there is no 001 string pattern available, then cut down the first 0, and check for only 01, no there is no 01 string pattern available, then cut down the first 0, and check for only 1, yes the 1 pattern is available in the string state is q2. So $\delta: q3 \times 1 \rightarrow q1$

Q/ Σ	0	1	
q0	q0	q1	----- ϵ
q1	q2	q1	-----1
q2	q3	q1	-----10
q3	q0	q1	----100

Now the transition table solving is complete based on string divide and matching pattern strategy and the next step is to design the state transition diagram for Deterministic Finite Automata(DFA).

Initially we draw the states and make transitions, the mathematical representation for DFA is,

$M = (Q, \Sigma, \delta, q_0, F)$

$Q - q_0, q_1, q_2, q_3$

$\Sigma - \{0,1\}$

$q_0 - q_0$

$F - q_3$.

δ - Transition Function for every state

$\delta: q_0 \times 0 \rightarrow q_0$

$\delta: q_0 \times 1 \rightarrow q_1$

$\delta: q_1 \times 0 \rightarrow q_2$

$\delta: q_1 \times 1 \rightarrow q_1$

$\delta: q_2 \times 0 \rightarrow q_3$

$\delta: q_2 \times 1 \rightarrow q_1$

$\delta: q_3 \times 0 \rightarrow q_0$

$\delta: q_3 \times 1 \rightarrow q_1$

Then we will draw state transition diagram for above transition table. In the state transition diagram the states can be representing with the circle symbol and transitions can be representing using arrow mark. Within that arrow mark we will specify the input symbol. Based on the arrow mark it will show the transition function the current state with an input symbol moves to next state.

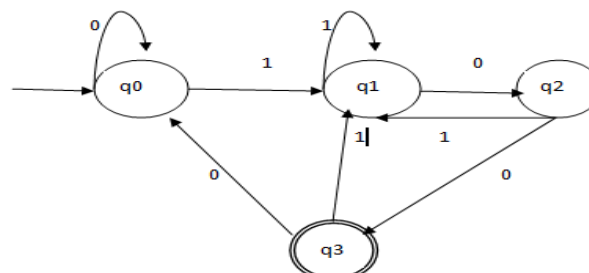


Figure: State transition diagram

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

After designing of a DFA, check whether the automaton is valid or not. To check the DFA is valid take any of the string and check whether it is accepted or not.

Example: string 1100

The string can be start from initial or start state and check the transitions for the string. At the end of the string if the machine/ automata is in accept or final state then the automata/DFA is valid for the given language. Otherwise it not valid for the given language condition.

Transition funtion δ : $\delta(q_0, 1100) = \delta(\delta(q_0, 1), 100)$
 $= \delta(\delta(q_1, 1), 00)$
 $= \delta(\delta(q_1, 0), 0)$
 $= \delta(\delta(q_2, 0), 0)$
 $= q_3$ i.e Final state

After string verification the machine resides in final/accepts state. So the machine is valid for the given condition.

IV. CONCLUSION

This paper proposes the strategy to design an automaton which used to compute a specific problem. And that strategy is different from general procedure which is easy to design and traditional methodology based on patterns generated from the input string. And that input string is from 0th string to nth string from input and matching every step of pattern to make transitions for every transitions for every state. And the states also decided based on input string only. This strategy is a robust solving method for designing an automaton, which is easy and more strategic way of solving a problem or design an automaton.

REFERENCES

- [1] AnatBremmler Barr, D.Hay, Y. Koral, —CompactDFA: Scalable pattern matching Using Longest Prefix Match Solutions," in IEEE/ACM Transaction on networking, vol 22, No.2, April 2014.
- [2] A.V. Aho and M. J. Corasick. —Efficient String Matching: An Aid to Bibliographic Search. I Communications of the ACM, 18(6):333 340, 1975.
- [3] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, —Algorithms to accelerate multiple regular expressions matching for deep packet inspection, in Proc. of ACM SIGCOMM , pages 339 350. ACM, 2006.
- [4] S. Kumar, J. Turner, J. Williams, —Advanced algorithms for fast and scalable deep packet inspection, in Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS), pages 81 - 92. ACM, 2006.
- [5] M. Becchi, P. Crowley, —A hybrid finite automaton for practical deep packet inspection, in Proc. Conf. Emerging Netw. Exp. Technol. (CoNEXT), pages 1 - 12, 2007.
- [6] S. Kumar, B. Chandrasekaran, J. Turner, G. Varghese, —Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia, in Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS), pages 155- 164. ACM, 2007.
- [7] R. Smith, C. Estan, and S. Jha, —Xfa: Faster signature matching with extended automata, in IEEE Symposium on Security and Privacy, May 2008.
- [8] D.Ficara, S.Giordano, G. Procissi, F.Vitucci, G.Antichi, A.D. Pietro, —An Improved DFA for Fast Regular Expression Matching, ACM SIGCOMM Computer Communication Review, Volume 38, Number 5, October 2008.
- [9] S. K ong, R. Smith, and C. Estan, —Efficient signature matching with multiple alphabet compression tables, in Proc. Int. Conf. Security Privacy Commun. Netw. (Securecomm), 2008.
- [10] S. Kumar, J. Turner, P. Crowley, and M. Mitzenmacher, —HEXA: Compact data structures for faster packet processing", in Proc. IEEE Conf. Comput. Commun. (INFOCOM), 2009. Ms. Utkarsha P. Pisolkar, Asst. Prof. Shivaji R. Lahane, A Survey on Deterministic Finite Automata Compression Techniques, ISSN: 2278 – 1323, 2015 IJARCET



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)