ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **9** (2020), no.9, 6971–6980
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.9.9.46

ALGORITHM FOR CONSTRUCTING DETERMINISTIC FINITE AUTOMATA DIRECTLY FROM REGULAR EXPRESSIONS FOR PATTERN RECOGNITION RELATED PROBLEMS

M. P. RAJAKUMAR[1], J. RAMYA, R. SONIA, AND B. UMA MAHESWARI

ABSTRACT. Theory of computation, a division of theoretical computer science deals with how proficiently problems can be explained on a model of computation, by means of an algorithm. Automata theory play a key part in theory of computation, design of compiler and formal verification. An automaton can be designed to perform a variety of tasks related to various domains of human life such as language recognition process. The automata theory has three main classes that include the deterministic finite automata (DFA), nondeterministic finite automata (NFA) and the nondeterministic finite automata with Epsilon (NFA- $\mathcal{E}$). In this paper a novel algorithm for constructing DFA directly from regular expressions (REs) for pattern recognition problems are proposed. This algorithm can be extended to accept more input variables with minor modifications.

1. INTRODUCTION

Theory of computation has three branches, namely computational complexity theory, computability theory and automaton theory. Automata theory is the study of abstract machines and automata. Finite state automata, pushdown automata and Turing machine are well known type of automata theory. This automaton consists of states (symbolized by circles) and transitions (represented

[1]*corresponding author*

2010 *Mathematics Subject Classification.* 68Q45.

Key words and phrases. Theory of computation, automata theory, deterministic finite automata, regular expression, pattern recognition problems.

by arrows). Automaton takes two arguments namely the state and input symbol and produces state as the output function, agreeing to its transition function.

Finite state machine has three main classes that include the DFA, NFA and the NFA- $\mathcal{E}$. Representing the transitions of DFA can be done using transition diagram and/or transition table. In a transition diagram, if (p, a) = q then the arrow goes from the vertex which corresponds to state p, to the vertex that corresponds to state q labeled a. In a transition table, rows corresponds to states and columns correspond to inputs. An entry corresponds to next states to indicate the transition of DFA.

The article is organized as follows. Section 2 provides the applications of Automata Theory and related works. We establish representation by concisely paraphrasing textbook definitions of automata in Section 3. Then, in Section 4, we introduce the proposed algorithm for constructing DFA that involves only two input variables. Final part of section 4 extended the proposed algorithm for construction pattern matching algorithm for 3 variables with minor modification in the original algorithm followed by the transition table involving four input symbols. Section 5 concludes the work.

2. Related Works

Finite Automata along with transducers have many interesting applications in image manipulation, image recognition and in language processing, etc. [1, 2, 3, 4, 5]. It can be also used in both loss and lossless compression [6]. The concept of automata is used to identify the learning disability of the students and provide the solutions to deal the disability [7]. They can be used in software testing as a tracking system [8]. They are very good in splicing system work [9].

The minimal DFA is constructed from RE using set of grammar rules [10]. Algorithm is developed to construct minimal DFA based on backward information [11]. To accelerate the generation of DFA, two different data structures were developed [12]. Even simple DFA constructed using some Algorithmic design techniques such as divide and conquer [13] and incremental algorithm [14]. Multidimensional FA based RE matching algorithm were proposed [15].

3. Notations and Basic Definitions

Definition 3.1 (Deterministic Finite Automata (DFA)). *Formally a DFA consists of 5-tuples M = (Q, Σ, δ, q_0, F) where finite set of states (Q), finite set of input symbols called alphabet (Σ), transition function δ defined as: Q X Σ $\rightarrow$ Q, initial state ($q_0 \in$ Q), set of accepting states (F $\subseteq$ Q) .*

Definition 3.2 (Extended transition function of DFA). *For DFA, M = (Q, Σ, δ, q_0, F) the transition function is extended as δ: Q X Σ*$\rightarrow$ Q and is defined as follows.*

- *For any state q of Q δ(q, $\mathcal{E}$) = q*
- *For any state q of Q, any string x $\in$ Σ* with 'a' as the last symbol of x and a $\in$ Σ , then δ(q, xa)= δ(δ(q, x),a).*

Definition 3.3 (Languages accepted by DFA). *The language accepted by DFA M = (Q, Σ, δ, q_0, F) is the set of strings Σ accepted by M i.e., L (M) = {w $\in$ Σ* / δ (q_0, w) is in F}.*

Definition 3.4 (Nondeterministic Finite Automata (NFA)). *Same as DFA except that the transition function δ defined as: Q X Σ $\rightarrow 2^Q$.*

Definition 3.5 (Regular Expression). *The languages acknowledged by finite automata are straight forwardly termed by simple expression called regular expression.*

4. Proposed Method

Although it is easier to check the participation in DFA, constructing DFA is more challenging task compared to NFA since in DFA state transition is limited it cannot use empty string transition. For the RE containing strings of a's and b's that starting with$(a + b)^*$ followed by aa, ab, ba or bb that is $(a + b)^*aa$, $(a + b)^*bb, (a + b)^*ab, (a + b)^*ba$ construction of NFA is easy. This is shown in the following Figure 1.

Likewise constructing DFA that starts with aa, bb, ab and ba followed by (a+b)* looks easy (Figure 2) since there is a restriction in only initial part of the string whereas construction of DFA to accept set of all string of a's and b's and end with ab, ba, aa or bb looks difficult.

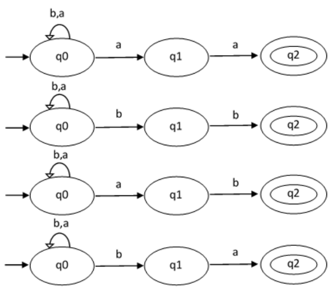


FIGURE 1. NFA for the RE $(a+b)^*aa, (a+b)^*bb, (a+b)^*ab, (a+b)^*ba$

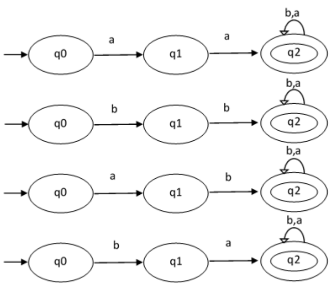


FIGURE 2. DFA for the RE $aa(a+b)^*, bb(a+b)^*, ab(a+b)^*, ba(a+b)^*$

The proposed method provides simple algorithm for the problem that involve pattern recognition. i.e. it constructs DFA for a given RE consists of strings of a's and b's and end with aa, bb, ab or ba. The procedure for constructing DFA is given in Algorithm 1. First construct the skeleton of the DFA. For this we have to identify the minimum length string accepted by the given RE. Then identify the finite set of alphabets for the DFA. For example, for the problem of DFA to accept the strings of a's, b's and c's having atleast one b and one c , the minimum string is b,c and the finite set of input alphabet is a, b,c.

Algorithm 1 : Construction of DFA for 2 input symbols

Input: RE (strings of a's and b's and end with aa, bb, ab or ba)
Output: The corresponding DFA.

Method

1.Base case: Construct the skeleton of the DFA by considering the minimum length of string.

2.Categorization

- **Categorization of transitions:** Categorize the transitions into defined transitions(DI) and not defined transition(NDI) in the base case.
- **Categorization of states:** Categorize the states into start state, intermediate state and accepting or final state from the skeleton of the DFA.
- **Categorization of input symbols:** Categorize the input symbol into ending input symbol (EIS) and non-ending input symbol (NIS) from the skeleton of the DFA.

3.Transition rules: (for 2 input alphabets)

- If **defined transitions** contain **same input** symbol
- Mark the following transitions on NIS
- From start state to **start state** (self-loop)
- From intermediate state to **start state**
- From final state to **start state**
- Mark the following transitions on EIS
- From final state to **final state** (self-loop)
- else (if **defined transitions** contain **different input** symbol)
- Mark the following transitions on NIS
- From intermediate state to **intermediate state**
- From final state to **intermediate state**
- Mark the following transitions on EIS
- From start state to **start state** (self-loop)
- From final state to **start state**

Next, classification is done based on transitions δ, states Q and input symbols Σ. The transitions is classified into two types namely DI and NDI. The DI contains the transitions defined in the skeleton of DFA and NDI can be easily identified by considering the number of states and number of input symbols. For example, the DI for the above example is δ (q_0, b) $=q_1$, δ (q_1, c) $=q_2$ and NDI are δ (q_0, a), δ (q_0, c),δ (q_1, a), δ (q_1, b), δ (q_2, a), δ (q_2, b) and δ (q_2, c). States are categorized into start state, intermediate state and accepting or final state. Here start state will be q_o, intermediate state is q_1 and set of accepting state is q_2. Categorization also done based on ending input symbol (EIS) and non-ending input symbol (NIS). EIS is the input symbol that ends with the final state and NIS is the remaining input symbols. For this case, EIS is c and NIS is a and b.

Transitions rules are framed based on the equivalence between DI contains equal symbol are not. If the DI contain the same input symbol then mark the transition on NIS from all states to start state and on EIS from final state to final state. If the DI contain the different input symbol then mark the transition on NIS from intermediate and final state to intermediate state and on EIS from start and final state to start state. Table 1 shows the construction steps of DFA from RE for all four possibilities.

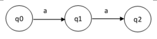
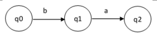
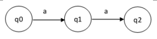
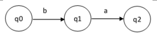
RE	(a+b)*aa	(a+b)*bb	(a+b)*ab	(a+b)*ba
Skeleton of RE				
DI	$\delta(q_0,\text{a})=q_1$ $\delta(q_1,\text{a})=q_2$	$\delta(q_0,\text{b})=q_1$ $\delta(q_1,\text{b})=q_2$	$\delta(q_0,\text{a})=q_1$ $\delta(q_1,\text{b})=q_2$	$\delta(q_0,\text{b})=q_1$ $\delta(q_1,\text{a})=q_2$
NDI	$\delta(q_0,\text{b})$, $\delta(q_1,\text{b})$ $\delta(q_2,\text{a})$, $\delta(q_2,\text{b})$	$\delta(q_0,\text{a})$, $\delta(q_1,\text{a})$ $\delta(q_2,\text{a})$, $\delta(q_2,\text{b})$	$\delta(q_0,\text{b})$, $\delta(q_1,\text{a})$ $\delta(q_2,\text{a})$, $\delta(q_2,\text{b})$	$\delta(q_0,\text{a})$, $\delta(q_1,\text{b})$ $\delta(q_2,\text{a})$, $\delta(q_2,\text{b})$
Start Intermdiate Final	q_0 q_1 q_2	q_0 q_1 q_2	q_0 q_1 q_2	q_0 q_1 q_2
EIS NIS	a b	b a	b a	a b
Transition on NIS	$\delta(q_0,\text{b})=q_0$ $\delta(q_1,\text{b})=q_0$ $\delta(q_2,\text{b})=q_0$	$\delta(q_0,\text{a})=q_0$ $\delta(q_0,\text{a})=q_0$ $\delta(q_0,\text{a})=q_0$	$\delta(q_1,\text{a})=q_1$ $\delta(q_2,\text{a})=q_1$	$\delta(q_1,\text{b})=q_1$ $\delta(q_2,\text{b})=q_1$
Transition on EIS	$\delta(q_2,\text{a})=q_2$	$\delta(q_2,\text{b})=q_2$	$\delta(q_0,\text{b})=q_0$ $\delta(q_2,\text{b})=q_0$	$\delta(q_0,\text{a})=q_0$ $\delta(q_2,\text{a})=q_0$

TABLE 1. Construction steps of DFA

The final DFA for the RE constructed based on the above algorithm is shown in Figure 3.

This RE to DFA conversion algorithm can be extended to three input variables involving a, b and c given in Algorithm 2. In this case there will be minor changes in the transition rules for DI that contain same input symbols and DI contains different symbols. For the DI containing the same input symbol, instead

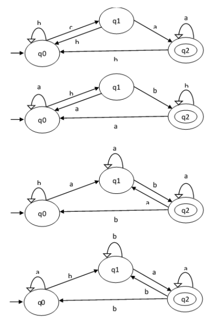


FIGURE 3. DFA for the regular expression $(a+b)*aa, (a+b)*bb, (a+b)*ab, (a+b)*ba$

of one NIS there will be two NIS namely NIS1 and NIS2 and two intermediate states namely intermediate state1 and intermediate state2 . For the case two, mark the transition on NIS1 from all states (except start state) to intermediate state1, on NIS2 from all states (except intermediate state1) to start state and on EIS from all states (except intermediate state2) to final state.

Algorithm 2 : Construction of DFA for 3 input symbols

If **defined transitions** contain **same input** symbol

- Mark the following transitions on NIS_1 and NIS_2
- From start state to **start state** (self-loop)
- From intermediates state to **start state**
- From final state to **start state**
- Mark the following transitions on EIS
- From final state to **final state** (self-loop)
- **else if**
- **Defined transitions** contain **different input** symbol
- Mark the following transitions on NIS$_1$
- From intermediate state1 to intermediate state1
- From intermediate state2 to intermediate state1
- From final state to intermediate state1
- Mark the following transitions on NIS$_2$
- From start state to start state
- From intermediate state2 to start state
- From final state to start state
- Mark the following transitions on EIS

- From start state to start state (self-loop)
- From intermediate state1 to start state
- From final state to start state

If there are 'n' input symbol then there will be 'n' DI and n^2 NDI. So the total number of transitions for 'n' input symbol is $n+n^2$. The transitions for the RE involving 4 input symbol is shown in the following transition table 2 and 3. In general, for 'n' input symbol there will be 'n-1' NIS (namely NIS_1 , NIS_2 .. $NIS_n - 1$) and one EIS will be available. All 'n-1' NIS has the same property for DI that contain the same symbol. All states (Starting state, Intermediate states and final state) has transitions to starting state on these $NIS_n - 1$ for DI that contain different input symbol, all states except starting state has transitions on NIS_1 to starting, state, the same symbol and n^2 NDI. So the total number of transitions for 'n' input symbol is $n+n^2$.

Input	NIS_1 , NIS_2 and NIS_3					EIS				
States	SS	IS_1	IS_2	IS_3	FS	SS	IS_1	IS_2	IS_3	FS
SS	✓									
IS_1	✓									
IS_2	✓									
IS_3	✓									
FS	✓									✓

TABLE 2. Transition table with DI contain same symbol

5. CONCLUSION

In this paper we established that, for a pattern recognition problem it is possible to build least DFA directly from RE, that is, without the intermediate lengthy chain conversion step of RE to NFA with epsilon, NFA with epsilon to NFA and NFA to DFA using predefined transition rules for which involves only 2 input symbols a's and b's. We also showed that construction algorithm can be extended to accept three input symbols a's, b's and c's with no change in the transition rules for DI that contain same input symbols and require minimal changes

Input	NIS_1					NIS_2					NIS_3					EIS				
States	SS	IS_1	IS_2	IS_3	FS	SS	IS_1	IS_2	IS_3	FS	SS	IS_1	IS_2	IS_3	FS	SS	IS_1	IS_2	IS_3	FS
SS						✓					✓					✓				
IS_1		✓									✓					✓				
IS_2		✓				✓														
IS_2		✓				✓					✓									
FS		✓				✓					✓					✓				

TABLE 3. Transition table with DI contain different symbol

in the transition rules for transitions that is not defined. Using this algorithm it is also possible to design DFA to accept strings of a's and b's which do not end with the string ab, ba, aa and bb. In future this algorithm can be extended to accept strings of input symbols to accept the substring related problems. This algorithm can be used as base for other problems such as divisible by k problems and module k counter problems.

REFERENCES

[1] M. MINDEK, M. BURDA: *Image storage, indexing and recognition with finite state automata*, International Journal of Computer Science, **33**(1) (2007), 1-5.

[2] F. KATRITZE, W. MERZENICH, M. THOMAS: *Enhancement of partitioning techniques for Image Compression using weighted finite automata*, Theoretical Computer Science, **313**(1) (2004), 133-143.

[3] S. BADER, S. HOLLDOBLER, A. SCALZITTI: *Semiring Artifial Neural Networks and weighted Automata – And an application to Digital Image Encoding, Lecture Notes in Computer Science*, Springer – verlag, Berlin, **32-38**(1) (2004), 281-294.

[4] J. KARRI: *Image processing using Finite Automata*, Studies in Computational Intelligence, **25** (2006), 171-208.

[5] Y. COTEN-SYGAI, S. WINTNER: *Finite-state Registered Automata and their uses in Natural languages*, Conference Proceedings of Finite-State methods and Natural language processing, (2005), 43-45.

[6] X. MA, H. CHEN: *Compression method based on Generalized Finite Automata*, Conference on Audio, Language an Image processing, (2008), 1688-1692.

[7] S. A. ALI, S. SOOMRO, A. G. MERIN, A. BAQI: *Implementation of Automata theory to improve the learning disability*, Sindh university Research Journal., **1** (2013), 193-196.

[8] T. E. Shugla, W. A. Ivanov, N. S. Vagarina: *Developing a software system for automate based code generation,*, Programming and Computer Software, **42** (2016), 167-173.

[9] S. H. Khairuddin, M. A. Ahmad, A. Adzhar: *Splicing System in Automata Theory: A Review*, Journal of Physics Conference Series, **1336** (2019), 1-11.

[10] S. Bhargava, G. N. Purohit: *Construction of a Minimal Deterministic Finite Automation for a Regular Expression*, International Journal of Computer Applications, **15**(4) (2011), 16-17.

[11] D. Liv, Z. Huang, Y. Zhang, X. Guo, S. Su: *Efficient Deterministic Finite Automata minimization Backward depth Information*, Plos One, **11**(11) (2016), 1-17.

[12] C. Xu, J.Su, Shuhuichen: *Efficient Exploring Efficient NFA data structures to Accelerate DFA generation*, Conference on Cryptography, Security and Privacy, (2018), 56-61.

[13] D. D. Ruikar, R. S. Hegadi: *Simple DFA Construction Algorithm using Divide and Conquer Approach*, Lecture Notes in Networks and Systems, **43** (2018), 245-255.

[14] C. Campeano, A. Paun, J. R. Smith: *An Incremental Algorithm for Constructing minimal Deterministic finite cover Automata*, Lecture Notes in Computer Science, **3845** (2005), 90-103.

[15] Y. Gong, G. Liu: *Image A novel regular expression matching algorithm based on multidimensional finite Automata*, Conference on high performance switching and Routing, (2014), 90-97.

Department Of Computer Science And Engineering
St. Joseph's College of Engineering
Chennai - 600119 Tamilnadu, India
Email address: rajranjhu@gmail.com

Department of Computer Science and Engineering
St. Joseph's College of Engineering
Chennai - 600119 Tamilnadu, India
Email address: ramsharsha@gmail.com

Department of Computer Science and Engineering
Jeppiaar SRR Engineering College
Chennai - 600119 Tamilnadu, India
Email address: sonia.j25@gmail.com

Department of Computer Science and Engineering
St. Joseph's College of Engineering
Chennai - 600119 Tamilnadu, India
Email address: mahespal2002@gmail.com