Conversion of Fuzzy Regular Expressions to Fuzzy Automata using the Follow Automata

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Rahul Kumar Singh
801231021

Under the supervision of:
Dr. Ajay Kumar
Assistant Professor
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"*Conversion of Fuzzy Regular Expressions to Fuzzy Automata using the Follow Automata*"** in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Ajay Kumar*** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Rahul Kumar Singh)
801231021

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Ajay Kumar)
Assistant Professor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

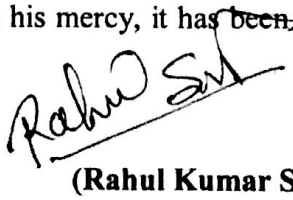
It is a great privilege to express my gratitude and admiration towards my respected supervisor **Dr. Ajay Kumar,** Assistant Professor, Computer Science & Engineering Department. He has been an esteemed guide and great support behind achieving this task. This work would not have been possible without the encouragement and guidance of my supervisor. His enthusiasm and optimism made this experience both rewarding and enjoyable. I am truly grateful to him for extending his total co-operation and understanding whenever I needed help and guidance from him.

I am also heartily thankful to **Dr. Deepak Garg**, Associate Professor and Head, Computer Science & Engineering Department and **Dr. Damandeep Kaur**, PG coordinator, for motivation and providing uncanny guidance and support throughout the preparation of the thesis.

I would also like to thankful to entire faculty and staff members of computer science & Engineering Department for their direct and indirect help, co-operation and affection which made my stay at Thapar University memorable.

Most Importantly, I would like to thanks my parents, my brother and my friends for showing me right direction and help me stay calm in the oddest of the times and keep moving even at time when there was no hope.

Last but not least, I would like to thankful to God for his blessing, showing me the right direction, not letting me down at the time of crises and with his mercy, it has been made possible for me to reach so far.



(Rahul Kumar Singh)

801231021

Abstract

Classical automata theory can not deal with uncertainty. To deal with system uncertainty, finite automata have been generalized into fuzzy automata. Stamenkovic and Ciric proposed an approach using the position automata for the construction of fuzzy automata from fuzzy regular expressions. There exist multifarious methodologies for the construction of finite automata from regular expressions known as Glushkov's position automata, follow automata, Antimirov partial derivatives and Thompson construction etc. Antimirov's partial derivative automata and Ilie's follow automata are the quotient of the Glushkov's position automata.

For the construction of fuzzy automata from regular expressions, if position automata is used then numbers of states are always n + 1 (where n is total count of input alphabets that are presented in the regular expression) and whereas if follow automata is used then numbers of states are equal to or less than n + 1 (where n is total count of input alphabets that are presented in the regular expression).

In this thesis, an approach for the conversion of fuzzy regular expressions to fuzzy automata using the concept of follow automata has been proposed. The number of states of the obtained Fuzzy automata using the proposed approach is lesser than the extant approaches in the literature without increasing the complexity. Using theorem, it has been proved that the language accepted by the fuzzy regular expression and fuzzy automata are same.

Table of Contents

List of Figures

List of Tables

Notations

Notation	Name
$\vee$	Supremum
$\wedge$	Infimum
$\otimes$	Binary operator
ρ_r	Homomorphism mapping of fuzzy regular expression
$\in$	Belong to
$\subseteq$	Subset of
$\forall$	For all
ε - move	Null move

Chapter 1 Introduction

Automata theory is firmly similar to formal language theory. An Automaton is an abstract computational device. Computation devices are used to solve many problem such as description of programming language, design digital system, neural network, artificial intelligence, pattern recognition, string matching, searching, database, lexical analysis and many more applications. The best known computational devices are Turing machine, finite automata, linear bounded automata, and pushdown automata. These machines are represented by their languages and grammars. The mechanism behind the digital computer is automaton.

1.1 Finite Automata

Finite automata have a finite memory in term of states and perform some activities like token finding and lexical analysis etc. Finite automata have input tape, read only head and control unit. Input tape is divided into cells and each cell contains input alphabet. Control unit contains a finite number of states and explains the internal structure of finite automata.

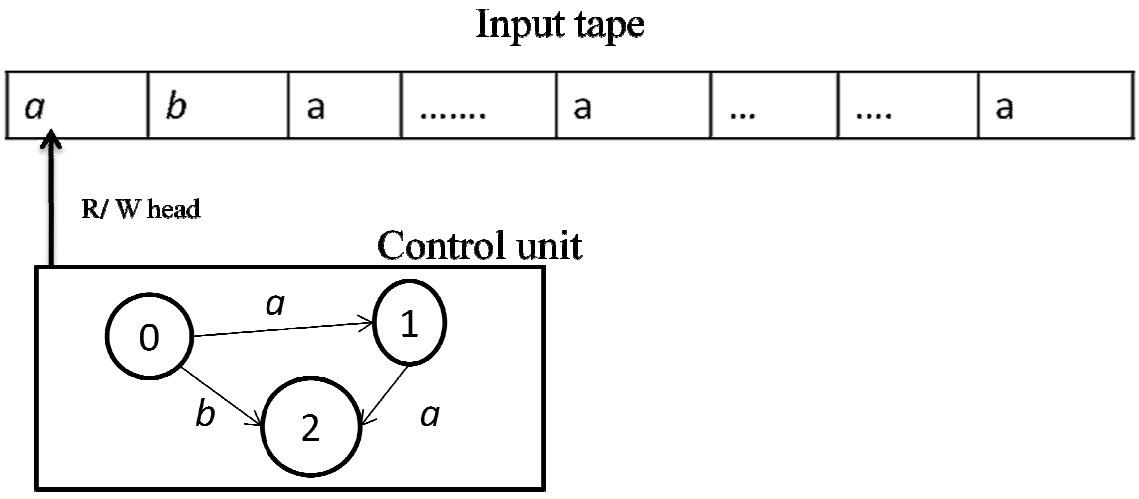


Figure 1.1: Finite automata

Read/ Write head, read the symbol from input tape and it can move only one way. It can not remember the previous input because finite automata do not have memory to remember. Finite automata have been classified into two types *i.e*. nondeterministic finite automata and deterministic finite automata.

1.1.1 Deterministic Finite Automata

Deterministic finite automata (DFA) [25] is a 5 – tuple M = (U, S, γ, u_0, B), where

i. U is a finite non – empty set of states.
ii. S is finite non – empty set of input alphabets.
iii. γ: $U \times S \rightarrow U$ is a transition function.
iv. $u_0 \in U$, where u_0 is initial state.
v. $B \subseteq U$ is set of final states.

In deterministic finite automata ε – moves are not allowed and each move is uniquely determined. Transition function γ for deterministic finite automata is explained as: γ (u_0, a) $\rightarrow u_1$, u_0 is current state. On reading input alphabet a in current state, reached in next state u_1. In deterministic finite automata, choice and dead configuration are not allowed. Deterministic finite language are closed under the complement, intersection, union and set difference.

The extended transition function of DFA is defined as: γ^*: $U \times S^* \rightarrow U$.

Languages accepted by the deterministic finite automata are explained as:

L (M) = $\{w \in S^* \mid \gamma^* (u_0, w) \in B\}$ where w is a string and B is final states.

Example 1.1.1 Consider the figure 1.2, deterministic finite automata M = (U, S, γ, u_0, B), where $U = \{u_0, u_1, u_2, u_3\}$, $S = \{a, b\}$, u_0 is a initial state, and $F = \{u_3\}$.

The transition function γ for string $w = aabb$ is defined in following way:

$$\begin{aligned}\gamma(q_0, w) &= \gamma^*(u_0, aabb),\\ &= \gamma^*(u_1, abb),\\ &= \gamma^*(u_1, bb),\\ &= \gamma\ (u_3, b),\end{aligned}$$

u_3 is a final state that means this string is accepted. But strings bbaa, aaba, aaaa are not accepted by this machine because these strings are not starting with a and not ending with b.

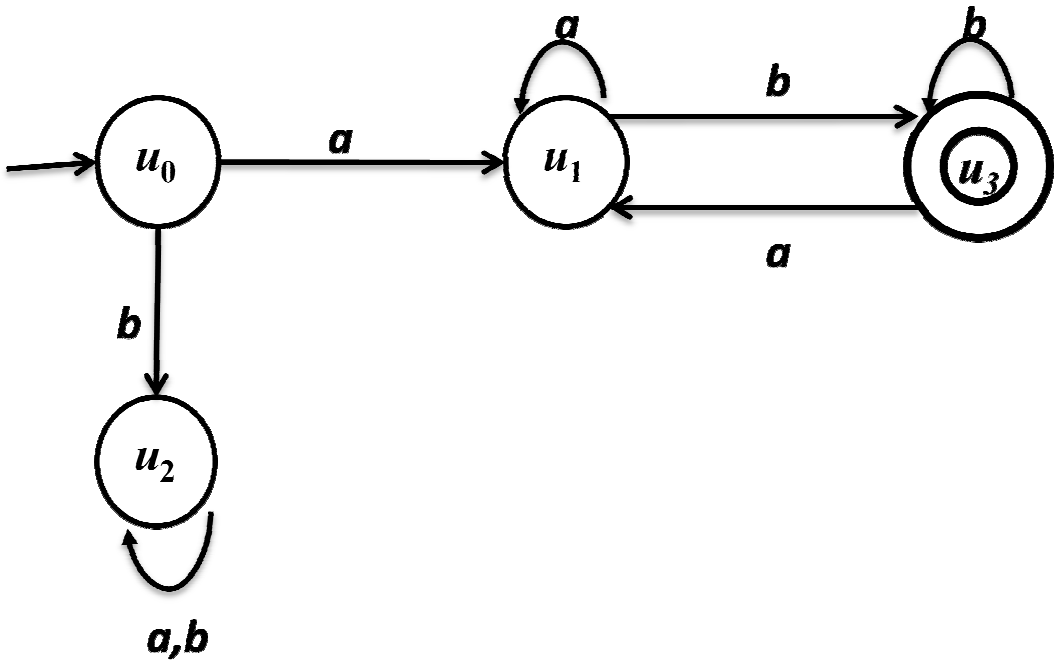


Figure 1.2: Deterministic finite automata

The language of this machine is defined as: L (M) = $\{w \in S^* \mid$ Starting with a and ending with $b\}$. In this machine every state is deterministic in nature or we can say there is no choice in state by reading same input alphabet.

1.1.2 Nondeterministic Finite Automata

Nondeterministic finite automata (NFA) [47] is a 5 – tuple M = (U, S, γ, u_0, B), where

i. U is a finite non – empty set of states.
ii. S is finite non – empty set of input alphabets.
iii. γ: $U \times (S \cup \{\varepsilon\}) \rightarrow 2^U$ is a transition function where 2^U is power set.
iv. $u_0 \in U$, where u_0 is initial state.
v. $B \subseteq U$ is set of final states.

In nondeterministic finite automata ε - moves allowed and each move may have choices. The mapping of transition function is defined as: $\gamma\ (u_0, a) \rightarrow \{u_1 \ldots\ldots u_n\}$.

The extended transition function for nondeterministic finite automata is γ^*: $U \times S^* \rightarrow 2^U$ or $\gamma(u_0, w, a) = \gamma(\gamma^*(q_0, w), a)$.

Language accepted by the nondeterministic finite automata is defined in following way:

L (M) = $\{w \in S^* \mid \gamma^* (u_0, w) \cap B \neq \emptyset\}$ where w is a string and B is final states.

Example 1.1.2 Consider the figure 1.3, NFA M = (U, S, γ, u_0, B) where $U = \{u_0, u_1, u_2, u_3\}$, $S = \{a, b\}$, u_0 is a initial state, and $B = \{u_3\}$.

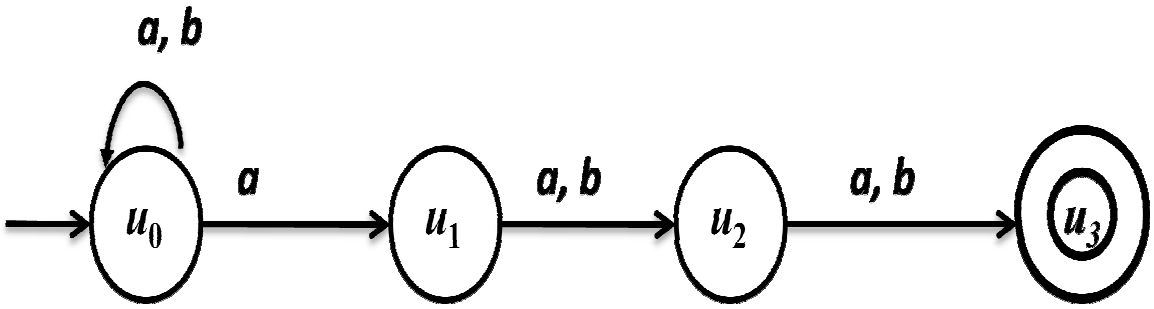


Figure 1.3: Nondeterministic finite automata

The transition function γ for string $w = aabb$ is defined as:

$$\gamma(u_0, w) = \gamma^*(q_0, aabb),$$

$$= \gamma^*(u_1, abb) \text{ or } \gamma^*(u_0, abb)$$

$$= \gamma^*(u_1, bb),$$

$$= \gamma\ (u_3, b),$$

In the above figure u_3 is a final state that means this string is accepted. But *bbaa*, *bbbb* and *baa* are not accepted by this machine. The language of this machine is defined as: L (M) = $\{w \in S^* \mid$ third symbol from right is $a\}$.

In the state u_0, machine read input alphabet a and reach in state u_0 or u_1. Hence in nondeterministic finite automata choice is allowed. The power of deterministic and nondeterministic finite automata is same. Nondeterministic finite automata are closed under intersection, union, complementation, set deference and kleene closure operation. We can convert nondeterministic finite automata to deterministic finite automata. Singh and Loura [42] carried out the comparison of various approaches for the conversion of deterministic finite automata to regular expressions. Ghuman and Kumar [16] proposed a new algorithm for the conversion of regular expression to union-free regular expression.

1.2 Position Automata

The concept of position automata M_P ($R.E$) has been introduced by Glushkov, McNaughton and Yamada in 1960's. Using position automata, we can construct the nondeterministic finite automata from regular expressions. In position automata number of state is always n + 1 (where n is the number of occurrence symbols in regular expression). Glushkov, McNaughton and Yamada defined the position automata in the following sets [15]:

I. $P_0(R.E) = \{ 0, 1, \ldots\ldots, |R.E|_S \}$,

II. $first(R.E) = \{ j \mid s_j w \in L(\overline{R.E}) \}$,

III. $last(R.E) = \{ j \mid w s_j \in L(\overline{R.E}) \}$,

IV. $follow(R.E,\ j) = \{ j \mid w s_k s_j u \in L(\overline{R.E}) \}, j > 0 \}$,

V. $follow(R.E,\ j) = (first(R.E))$,

VI.

$$last_0(R.E) = \begin{cases} last(R.E), & \varepsilon \notin L(R.E), \\ last(R.E) \cup \{0\}, & \varepsilon \in L(R.E). \end{cases}$$

Position automata M_p ($R.E$) is a five tuple (U^P, S, γ^p, 0, $B^{last_0\,(R.E)}$), where all the tuple are similar to the nondeterministic finite automata.

Example 1.2 Consider the regular expression $R.E = (0 + 1)\ (0^* + 10^* + 1^*)^*$ using position automata, we will construct the nondeterministic finite automata. Where set of states $U = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6\}$, set of input alphabets $S \in \{0, 1\}$, u_0 is an initial state and last is set of final states. Marked version of regular expression is $\overline{R.E} = (0_1 + 1_2)\ (0_3^* + 1_4 0_5^* + 1_6^*)^*$

First ($R.E$) = $\{1, 2\} = \{u_1, u_2\}$,

Last ($R.E$) = $\{1, 2, 3, 4, 5, 6\} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$,

Follow ($R.E$, 1) = $\{3, 4, 6\}$,

Follow ($R.E$, 2) = $\{3, 4, 6\}$,

Follow ($R.E$, 3) = $\{3, 4, 6\}$,

Follow (*R.E*, 4) = {3, 4, 5, 6},

Follow (*R.E*, 5) = {3, 4, 5, 6},

Follow (*R.E*, 6) = {3, 4. 6}.

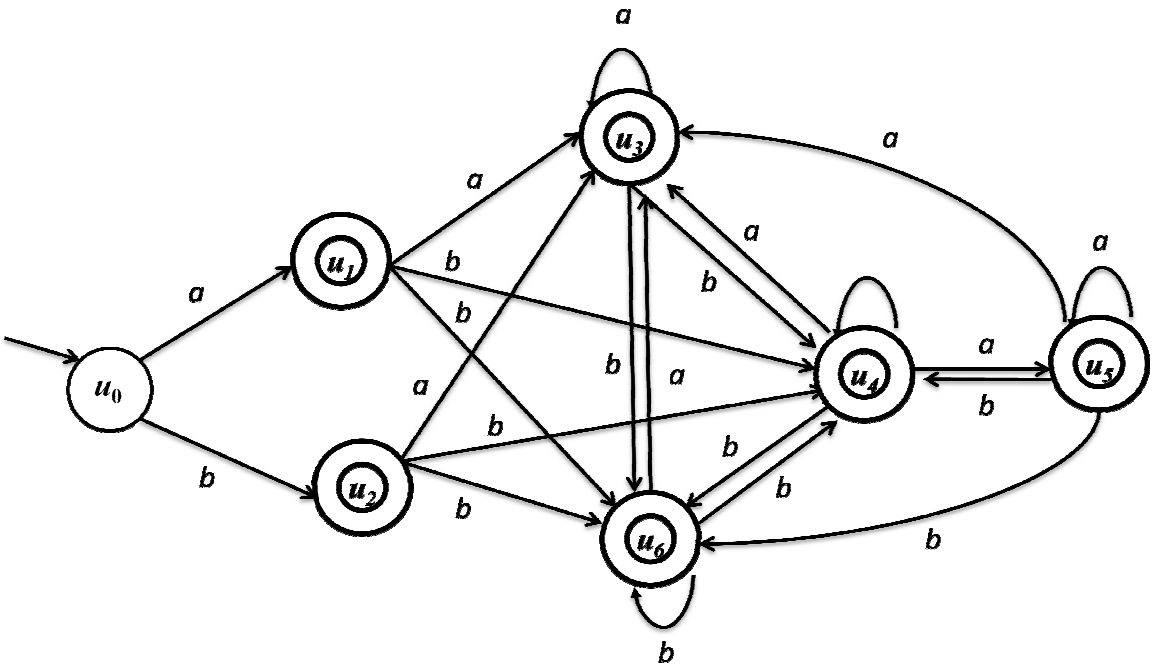


Figure 1.4: Position automaton $M_P(R.E)$ for $\overline{R.E} = (0_1 + 1_2)\,(0_3^* + 1_4 0_5^* + 1_6^*)^*$

We can observe that number of state in position automata is always n + 1 (where n is the number of occurrence symbols in regular expression.). In figure 1.4, number of states are 7 and position automaton is the best way to convert regular expression to nondeterministic finite automata.

1.3 Follow Automata

The concept of follow automata $M_F(R.E)$ has been proposed by Ilie and Yu in 2003 [19]. Follow automaton is used for construction of nondeterministic finite automata from regular expressions. Follow automaton is a quotient of position automata. Initially follow automata convert the regular expressions into smaller NFA, and then eliminate the ε-transitions. Follow automata $M_F(R.E)$ is a five tuple $(U^f, S, \gamma^f, u_0, B^f)$ where all the tuple are similar to the nondeterministic finite automata [19].

In starting we explained the equivalence $\equiv_f \subseteq p_0(R.E)^2$ by [19]

$p \equiv_f q$ if and only if (i) Follow ($R.E$, p) = Follow ($R.E$, q) and

(ii) Both p, q or none belong to last ($R.E$)

Using these two equations we can construct the follow automata.

Example 1.3 Consider the regular expression $R.E = (0 + 1)\ (0^* + 10^* + 1^*)^*$ using follow automata, we will construct the nondeterministic finite automata. Where $U = \{u_0, u_1, u_2\}$ is set of states, input alphabet $S \in \{0, 1\}$, u_0 is an initial state. Marked version of regular expression is $\overline{R.E} = (0_1 + 1_2)\ (0_3^* + 1_4 0_5^* + 1_6^*)^*$

First ($R.E$) = {1, 2},

Last ($R.E$) = {1, 2, 3, 4, 5, 6},

Follow ($R.E$, 1) = {3, 4. 6},

Follow ($R.E$, 2) = {3, 4, 6},

Follow ($R.E$, 3) = {3, 4, 6},

Follow ($R.E$, 4) = {3, 4, 5, 6},

Follow ($R.E$, 5) = {3, 4, 5, 6},

Follow ($R.E$, 6) = {3, 4. 6}.

Then the equivalence classes of follow automata are $\equiv_f$: {0} = $\{u_0\}$,

{1, 2, 3, 6} = $\{u_1\}$,

{4, 5} = $\{u_2\}$,

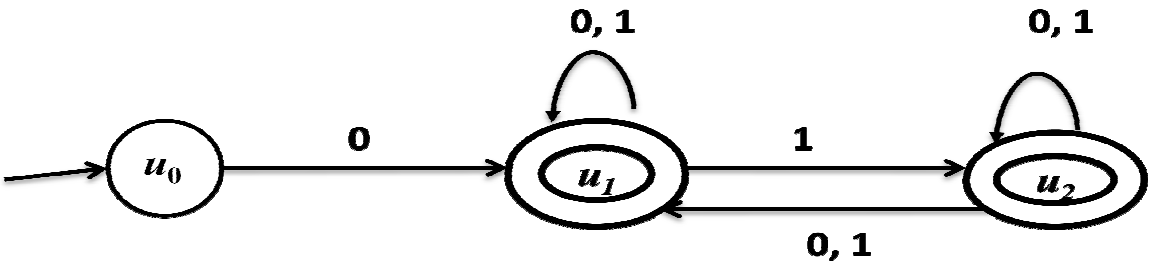


Figure 1.5: Follow automaton $M_F(R.E)$ for $\overline{R.E} = (0_1 + 1_2)\ (0_3^* + 1_4 0_5^* + 1_6^*)^*$

Figure 1.5 is the follow automata of the regular expression $R.E = (0_1 + 1_2)\ (0_3^* + 1_4 0_5^* + 1_6^*)^*$ and in follow automata number of states are always less than or equal to n + 1 (where n is the number of occurrence symbols in regular expression). In figures 1.5, number of states are 3 that is lesser than position automata. Researchers [20 - 22] extended the concept of followpos, firstpos and lastpos for the conversion of Parallel Regular Expressions to Non-deterministic Finite Automata.

1.4 Fuzzy Sets

In 1969 Zadeh's [24] introduced the Fuzzy sets. In crisp set, whether any object is the member or not, if object is the member of set then its membership value is 1 otherwise 0. But the fuzzy sets support flexibility in the sense of membership of object in a set and fuzzy set membership value or degree of membership lies between 0 and 1. Suppose fuzzy set P, membership function $\omega_P(\mathrm{k})$ and universal set or reference set K then function maps every element of reference set K in to the interval [0, 1]. The mathematical representation of mapping is as follows:

$$\omega_P(\mathrm{k}) : \mathrm{K} \rightarrow [0, 1]$$

A fuzzy set is explained in following way:

If K is a reference set and k is a one object or element of K, then a fuzzy set P defined on universal set K can be written as:

$$P = \{(\mathrm{k}, \omega_P(\mathrm{k})) \mid \mathrm{k} \in \mathrm{K}\}$$

According to the Zadeh's notation we can define fuzzy set as:

$$P = \sum_{k_i \in K} \omega_P\,(k_i)/k_i$$

Singleton notation is define like 1/ k where 1 is membership value of k in universal set K and 0 membership for all the other object of K.

1.4.1 Basic Operations of Fuzzy Sets

Given K to be the reference set, P and Q to be the two fuzzy sets and the membership function of set P and Q respectively $\omega_P(k)$ and $\omega_Q(k)$. The basic operations of fuzzy set

are union, intersection, complementation, scalar product etc. The following fuzzy sets operations are defined as:

1.4.1.1 Union

The union of two fuzzy sets P and Q is denoted as $P \cup Q$, and on the reference set K the membership function of the new fuzzy set is explained in the following way:

$$\omega_{P \cup Q}(k) = \max(\omega_P(k), \omega_Q(k)) \text{ or } \omega_P(k) \vee \omega_Q(k) \text{ where } \vee \text{ supremum.}$$

1.4.1.2 Intersection

The intersection of two fuzzy sets P and Q is denoted as $P \cap Q$, and on the reference set or universal set K the membership function of the new fuzzy set is explained in the following way:

$$\omega_{P \cap Q}(k) = \min(\omega_P(k), \omega_Q(k)) \text{ or } \omega_P(k) \wedge \omega_Q(k) \text{ where } \wedge \text{ infimum.}$$

1.4.1.3 Scalar Product

The scalar product $\lambda \cdot P$, where $\lambda \in [0, 1]$, P is fuzzy set and membership function of new scalar product is defined in following way:

$$\omega_{\lambda \cdot P}(k) = \lambda \cdot \omega_P(k),$$

These are the basic operation of fuzzy sets and there are some properties of fuzzy sets that are commutative, associative, distributive, idempotent, and identity. The composition of fuzzy relation is max – min and min – max composition is defined in following way: Given two relation A and B, and A to be relation on U, V and B to be a relation on V, W then $A \subseteq U \times V$ and $B \subseteq V \times W$. If the membership function of relation A and B is $\omega_A(u, v)$ and $\omega_B(v, w)$ respectively then min – max composition is defined in following way:

$$\omega_{A \circ B}(u, w) = \min[\max(\omega_A(u, v), \omega_B(v, w))].$$

Max – min composition is defined in following way:

$$\omega_{A \circ B}(u, w) = \max[\min(\omega_A(u, v), \omega_B(v, w))]$$

These min – max and max – min compositions are used in fuzzy automata for finding the fuzzy language of machine.

1.5 Lattice – Ordered Monoid

Fuzzy automata fully depend on membership value and in the recent year's researchers mainly focus on fuzzy automata with membership value in lattice – ordered monoid, complete lattice, complete residuated lattice and many different types of lattices. With the help of complete residuated lattice, fuzzy automata can take membership value was first introduced in [17] and many researchers carried their work [26, 43]. Fuzzy automata taking membership degree in lattice – ordered monoid is introduced in [27]. Fuzzy automata taking a membership degree over the different type of lattice is also defined by many researchers [26, 41, 43]. Lattice – ordered monoid is an algebraic structure, which contain infimum, supremum, binary operator, greatest element, least element and identity element i.e. L.M = (L, $\otimes$, $\vee$, $\wedge$, 1, 0, e).

1.5.1 Monoid

Monoid is a group which contains three properties that are closure property, associative property and identity property. (L, e, $\otimes$) is a monoid where L is a set, e is identity element, and $\otimes$ is a binary operation.

1.5.2 Lattice

A poset (L, $\leq$) is a lattice if and only if $\forall$ $a, b \in$ L, both least upper bound $\{a, b\}$ and greatest lower bound $\{a, b\}$ should be exist and belong to L. Given a lattice (L, $\vee$, $\wedge$, 1, 0) where L is lattice, $\vee$ to be supremum, infimum is $\wedge$, greatest and least element is 1 and 0.

1.5.3 Lattice – Ordered Monoid

Let (L, $\otimes$, $\vee$, $\wedge$, 1, 0, e) be a lattice – ordered monoid, if it is satisfied following two properties [41],

i. $\forall\ i \in$ L, $i \otimes 0 = 0 \otimes i$,

ii. $\forall\ i, j, k \in$ L, $i \otimes (j \vee k) = (i \otimes j) \vee (i \otimes k)$, and $(j \vee k) \otimes i = (j \otimes i) \vee (k \otimes i)$.

The binary operation $\otimes$ is multiplication.

1.5.4 Quantale

Let L.M = (L, $\otimes$, $\vee$, $\wedge$, 1, 0, e) be complete lattice and it satisfied the property of infinite distributive law that is define below [41]:

$$q \otimes \bigvee_{i \in I} q_i = \bigvee_{i \in I} (q \otimes q_i), \quad \bigvee_{i \in I} q_i \otimes q = \bigvee_{i \in I} (q_i \otimes q),$$

Then L.M is called as *quantale*. Lattice – ordered monoid called integral lattice – ordered monoid if the unit or identity element of monoid (L, e, $\otimes$) and the greatest element 1 of the lattice (L, $\vee$, $\wedge$, 1, 0) are coinciding. If integral lattice – ordered monoid with the property of commutative multiplication then it is called as complete residuated lattice.

1.6 Fuzzy Automata

Automata theory deals with certainty in any system modeling. Automata theory can not deal with fuzziness in any system. To deal fuzziness in any system, automata theory has been generalized into fuzzy automata theory. There are many applications where we can use fuzzy automata such as description of natural language and a programming language, lexical analysis, token finding, target recognition, fire detection, database, neural network, controlling system, learning system, pattern recognition, searching, discrete event system, fault detection, control system, image processing, artificial intelligence, monitoring system and many other area [34, 38, 41, 49].

To handle the system fuzziness, fuzzy automata is introduced by Santos in 1960's [40] and in 1969 Wee introduced the mathematical representation of fuzzy automata [48]. In 1969 Lee and Zadeh [24] proposed Fuzzy automata and languages over the input alphabet S. Some of the researchers explain that it similar to mealy machine and it has output capability. Many of the researchers describe fuzzy automata as deterministic fuzzy automata.

In the fuzzy automata, transition function plays an important role and categorized in the three different types. According to the first type the fuzzy transition function is defined as : $U \times S \rightarrow P(U)$ where U represents the set of states, S represents set of input alphabets, and $P(U)$ represents the set of all fuzzy subset of Q. Fuzzy transition function $\gamma : U \times S \rightarrow P(U)$ is equivalent to γ: $U \times S \times U \rightarrow [0, 1]$, where [0, 1] is an interval and we can replace this interval [0, 1] with lattice – ordered monoid or any other algebraic structure such as complete lattice, and complete residuated lattice etc. According to the second type of fuzzy transition function, the fuzzy transition function of fuzzy automata is

considered similar to the transition function of deterministic finite automata and captures the fuzziness in the fuzzy initial u_0 and final state b_0. According to the third type [8], improve the first type of fuzzy transition function or new component can add in the first type of fuzzy transition relation.

In fuzzy automata and languages, composition inference method can be used. Composition inference methods are max – product, min – max composition, max – min composition. With the help of these compositional inference methods we can find out the fuzzy language. In the classical automata string is either accepted and rejected but in fuzzy automata string is accepted with certain membership value. Similar to the classical automata theory fuzzy automata is also two types *i.e.* deterministic fuzzy automata and nondeterministic fuzzy automata.

1.6.1 Deterministic Fuzzy Automata

Deterministic fuzzy automata [40] is a 5 – tuple M = (U, S, γ, u_0, B), where

i. U is a finite non – empty set of states.
ii. S is finite non – empty set of input alphabets.
iii. γ: $U \times S \rightarrow P(U)$ is a fuzzy transition function.
iv. $u_0 \in U$, where u_0 is initial state.
v. $B \subseteq P(U)$ is fuzzy subset of final states.

Fuzzy transition function γ play important role in fuzzy automata. In state u_0, fuzzy transition function γ take input alphabet and reached in any state with certain membership value. For clarity, we can use the notation $u \xrightarrow{a} \omega$ to denote $\gamma(u, a) = \omega$. The fuzzy language denoted by S^* is set of all finite strings including empty stringε. The extended fuzzy transition function γ^* is explained in following way:

$$\gamma^*(u_0, wb) = \gamma(\gamma^*(u_0, w), b) = \bigcup_{u_1 \in U}\{ \gamma(u_0, w)(u_1) \cdot \gamma(u_1, b) \},$$

Where $u_0, u_1 \in U$, $\forall\ w \in S^*$ and $\forall\ b \in$ S, and 1/ u_0 is a singleton in U

Another way to represent the extended fuzzy transition function γ^* is defined in following way:

If u_0, $u_1 \in U$ and $\varepsilon \in S^*$ is the null word, then,

$$\gamma^{*}(u_0, \varepsilon, u_1) = \begin{cases} 1 & if\ u_0 = u_1, \\ 0 & otherwise, \end{cases}$$

And if $u_0, u_1 \in U$, $w \in \Sigma^{*}$ and $a \in \Sigma$ then,

$$\gamma * (u_0, wa, u_1) = \bigvee_{a \in S} \gamma * (u_0, w, u_2) \otimes \gamma(u_2, a, u_1).$$

The language accepted by machine L (M) with membership function is defined in following way:

$$\text{L (M)}(w) = \vee_{u \in U} \{ \text{ height}(\gamma^{*}(u_0, w) \wedge F) \}, \text{ where } w \in S^{*}.$$

The string w is accepted in fuzzy automata with certain membership value. For finding the language of strings in fuzzy automata, some approaches such as max – product automata, min – max automata and max – min automata can be used.

Example 1.6.1 Consider the figure 1.6, deterministic fuzzy automata M = (U, S, γ, u_0, B) where set of state $U = \{u_0, u_1, u_2\}$, set of input alphabet $S = \{a, b\}$, u_0 is a initial state, and final state $B = \{\frac{0.25}{u2}\}$.

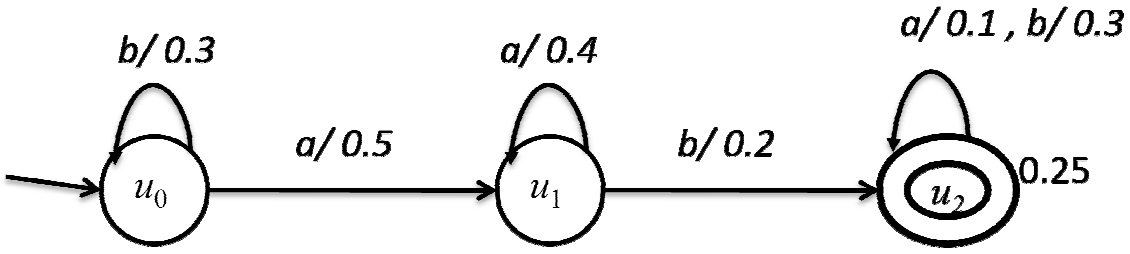


Figure 1.6: Deterministic fuzzy automata

The extended fuzzy transition γ^{*} is explained in following ways:

$$\gamma\ (u_0, \varepsilon) = \{\frac{1}{u0}\},$$

$$\gamma^{*}\ (u_0, ab) = \{\gamma^{*}(\gamma\ (u_0, a), b)\} = \{\ \gamma\ (u_0, a)\ .\ \gamma\ (u_1, b)\} = \{\frac{0.2}{u2}\},$$

$$\gamma^{*}\ (u_0, aabb) = \{\gamma^{*}(\gamma\ (u_0, a), abb)\} = \{\frac{0.2}{u2}\},$$

$\gamma^*(u_0, babab) = \{\gamma^*(\gamma(u_0, b), abab)\} = \{\frac{0.1}{u2}\}$.

With the help of extended fuzzy transition function, we can find out the membership value with respect to state. In the deterministic fuzzy automata, choices and null move are not allowed. Table 1.1 shows the fuzzy transition function for deterministic fuzzy automata at each state with input alphabet $S = \{a, b\}$.

Table 1.1: Fuzzy transitions function for deterministic fuzzy automata.

γ	a	b
u_0	$\{\frac{0.5}{u_1}\}$	$\{\frac{0.3}{u_0}\}$
u_1	$\{\frac{0.4}{u_1}\}$	$\{\frac{0.2}{u_2}\}$
u_2	$\{\frac{0.1}{u_2}\}$	$\{\frac{0.3}{u_2}\}$

Using max – min automaton, we can find out the fuzzy language for deterministic fuzzy automata.

$$L(M)(w) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, w) \wedge B) \}.$$

$$L(M)(\varepsilon) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, \varepsilon) \wedge B) \} = 0,$$

$$L(M)(ab) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, ab) \wedge B) \} = 0.2,$$

$$L(M)(aabb) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, aabb) \wedge B) \} = 0.2,$$

$$L(M)(babab) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, babab) \wedge B) \} = 0.1.$$

The membership value of strings ε, *ab*, *aabb*, and *babab* using max – min automata are respectively 0, 0.2, 0.2, and 0.1.

Using min – max automaton, we can find the fuzzy language of deterministic fuzzy automata.

$$L(M)(w) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, w) \vee B) \}.$$

$$L(M)(\varepsilon) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, \varepsilon) \vee B) \} = 0,$$

$$L(M)(ab) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, ab) \vee B) \} = 0.5,$$

$$L(M)(aabb) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, aabb) \vee B) \} = 0.5,$$

$$L(M)(babab) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, babab) \vee B) \} = 0.5.$$

The membership value of strings ε, *ab*, *aabb*, and *babab* using min – max automata are respectively 0, 0.5, 0.5, and 0.5.

1.6.2 Nondeterministic Fuzzy Automata

Nondeterministic fuzzy automata [8] is a 5 – tuple M = (U, S, γ, u_0, B), where

i. U is a finite non – empty set of states.
ii. S is finite non – empty set of input alphabets.
iii. γ: $U \times (S \cup \{\varepsilon\}) \rightarrow 2^{P(U)}$ is a fuzzy transition function, where $2^{P(U)}$ is a power set.
iv. $u_0 \in U$, where u_0 is initial state.
v. $B \subseteq P(U)$ is fuzzy subset of final states.

Fuzzy transition function γ in state *u,* takes an input from input tape and choose any one of the possibility distribution in $\gamma(u, a)$ with certain membership value. In this nondeterministic we cover both the case *i.e.* nondeterministic fuzzy automata with or without the null move. All the four tuple of nondeterministic fuzzy automata are same as deterministic fuzzy automata. The difference between deterministic and nondeterministic fuzzy automaton is in fuzzy transition function. The extended fuzzy transition function is defined in following way:

$$\gamma^*(u_0, wb) = \gamma(\gamma^*(u_0, w), b) = \bigcup_{u_1 \in U} \{ \gamma(u_0, w)(u_1) \cdot \gamma(u_1, b) \},$$

Where $u_0, u_1 \in U$, $\forall\ w \in S^*$ and $\forall\ b \in S$, and $1/u_0$ is a singleton in U

$$\gamma_\varepsilon(u, \varepsilon) = \{\frac{1}{u}\} \cup \gamma(u, \varepsilon),$$

$$\gamma_{\varepsilon}^{*}\ (u,\ \varepsilon^{n}\varepsilon) = \{\ \gamma^{*}\ (u,\ \varepsilon^{\mathrm{n}}).\ \gamma\ (u,\ \varepsilon)\},$$

Another way to represent the extended fuzzy transition function γ^{*} is defined in following way:

If $u_0, u_1 \in U$ and $\varepsilon \in S^{*}$ is the null word, then,

$$\gamma^{*}\ (u_0,\ \varepsilon,\ u_1) = \begin{cases} 1 & if\ u_0 = u_1, \\ 0 & otherwise, \end{cases}$$

And if $u_0, u_1 \in U$, $w \in \Sigma^{*}$ and $a \in \Sigma$, then

$$\gamma * (u_0, wa, u_1\ ) = \bigvee_{a\ \in \mathrm{S}} \gamma * (u_0, w, u_2\ ) \otimes \gamma(u_2, a, u_1).$$

The fuzzy language is accepted by machine based on max – min automaton is defined follows:

L (M)(w) = $\vee_{\ u\ \in U}$ {height($\gamma^{*}\ (u_0,\ w)\ \wedge\ B$)}, where $w \in S^{*}$. The string w is accepted in fuzzy automata with certain membership value.

Example 1.6.2 Consider the figure 1.7, nondeterministic fuzzy automata M = (U, S, γ, u_0, B) where set of state U = {u_0, u_1, u_2, u_3, u_4}, set of input alphabet S = {a, b}, u_0 is a initial state, and set of final state B = $\{\frac{0.4}{u_2} + \frac{0.8}{u_4}\}$.

The extended fuzzy transition function is defined in following way:

$$\gamma\ (u_0,\ \varepsilon) = \{\frac{1}{u0}\},$$

$$\gamma\ (u_0,\ a) = \{\gamma\ (u_0, \varepsilon)\ .\ \gamma\ (u_0, a)\} = \{\frac{0.8}{u_1} + \frac{0.3}{u_2}\},$$

$$\gamma\ (u_0,\ b) = \{\gamma\ (u_0, \varepsilon)\ .\ \gamma\ (u_0, b)\} = \{\frac{0.8}{u_3} + \frac{0.3}{u_2}\}$$

$$\gamma^{*}\ (u_0,\ aa) = \{\ \gamma^{*}\ (\gamma\ (u_0, a), a\} = \{\frac{0.1}{u_1} + \frac{0.6}{u_4}, \frac{0.3}{u_2}\}$$

$$\gamma^{*}\ (u_0,\ ab) = \{\ \gamma^{*}\ (\gamma\ (u_0, a), b\} = \{\frac{0.1}{u_4} + \frac{0.1}{u_2}\}.$$

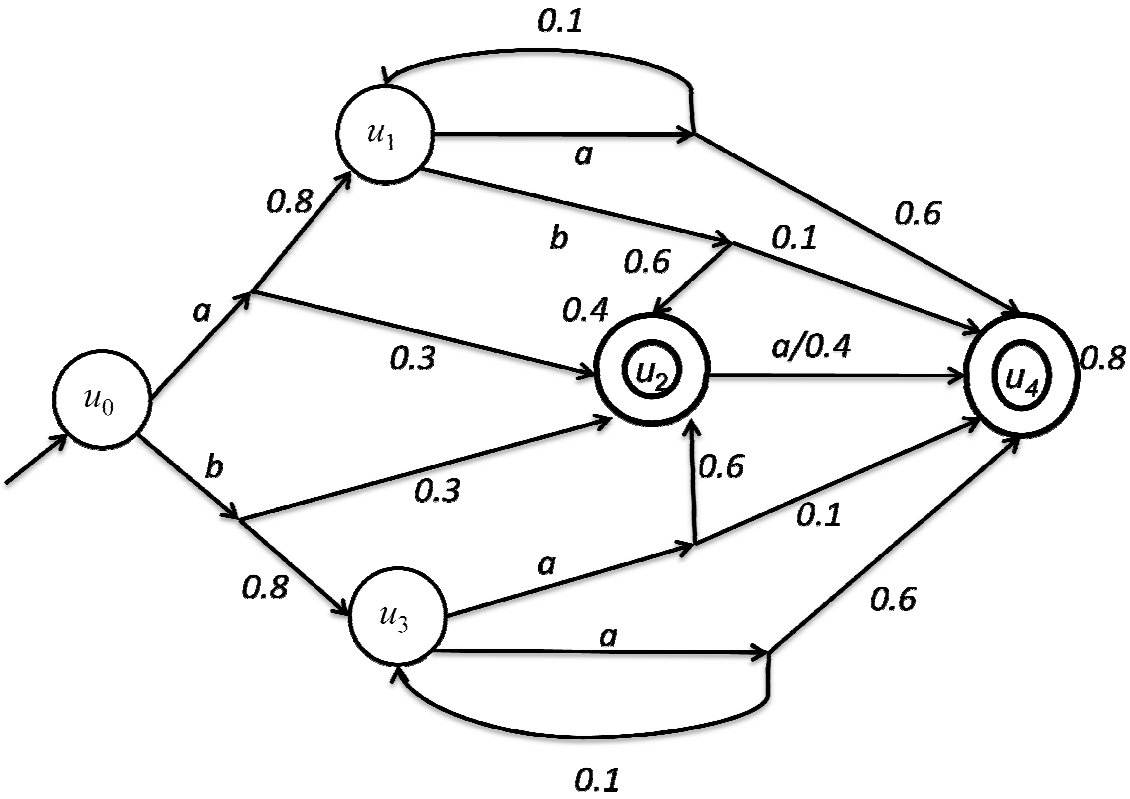


Figure 1.7: Nondeterministic fuzzy automata

Table 1.2 shows the fuzzy transition function for nondeterministic fuzzy automata at each state with input alphabet $S = \{a, b\}$.

Table 1.2: Fuzzy transition function for nondeterministic fuzzy automata

γ	a	b
u_0	$\{\frac{0.8}{u_1} + \frac{0.3}{u_2}\}$	$\{\frac{0.8}{u_3} + \frac{0.3}{u_2}\}$
u_1	$\{\frac{0.1}{u_1} + \frac{0.6}{u_4}\}$	$\{\frac{0.6}{u_2} + \frac{0.1}{u_4}\}$
u_2	$\{\frac{0.4}{u_4}\}$	ϕ
u_3	$\{\frac{0.6}{u_2} + \frac{0.1}{u_4}, \frac{0.1}{u_3} + \frac{0.6}{u_4}\}$	ϕ
u_4	ϕ	ϕ

Using max – min automaton, we find the fuzzy language of nondeterministic fuzzy automata. Fuzzy language of nondeterministic fuzzy automata over strings is explained in following way:

$$\text{L (M)}(w) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, w) \wedge B) \}.$$

$$\text{L (M)}(\varepsilon) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, \varepsilon) \wedge B) \} = 0,$$

$$\text{L (M)}(a) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, a) \wedge B) \} = 0.3$$

$$\text{L (M)}(b) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, b) \wedge B) \} = 0.3$$

$$\text{L (M)}(aa) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, aa) \wedge B) \} = 0.6$$

$$\text{L (M)}(ab) = \vee_{u \in U} \{ \text{height}(\gamma^*(u_0, ab) \wedge B) \} = 0.4.$$

Using min – max automaton, we find the fuzzy language of nondeterministic fuzzy automata. Fuzzy language of nondeterministic fuzzy automata over strings is explained in following way:

$$\text{L (M)}(w) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, w) \vee B) \}.$$

$$\text{L (M)}(\varepsilon) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, \varepsilon) \vee B) \} = 0,$$

$$\text{L (M)}(a) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, a) \vee B) \} = 0.8,$$

$$\text{L (M)}(b) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, b) \vee B) \} = 0.8,$$

$$\text{L (M)}(aa) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, aa) \vee B) \} = 0.4,$$

$$\text{L (M)}(ab) = \wedge_{u \in U} \{ \text{height}(\gamma^*(u_0, ab) \vee B) \} = 0.8.$$

In fuzzy automata, string is accepted with certain membership value. We can use different compositional methods such as max – product, min – max and max – min. Many researchers used the approach of max – min automaton for finding the fuzzy language. In this thesis, min – max automaton explained and using min – max automaton, find the fuzzy language of the fuzzy automata.

1.7 Fuzzy Regular Expression

Fuzzy regular expression (*FRE*) can used in many areas such string matching, pattern matching, database and searching. Let S be a set of finite non – empty set of input alphabet. Fuzzy regular expression r over the input alphabet S contains only seven steps that are explained below [27, 28]:

i. $\emptyset \in FRE$;
ii. $\varepsilon \in FRE$;
iii. $r \in FRE, \forall r \in S$;
iv. Scalar multiplication: $(\lambda r) \in FRE, \forall \lambda \in \mathrm{L}$ and $r \in$ R.E;
v. Addition: $(r_1 + r_2) \in FRE, \forall\, r_1, r_2 \in FRE$;
vi. Concatenation $(r_1\, r_2) \in FRE, \forall\, r_1, r_2 \in FRE$;
vii. Star operation $(r^*) \in FRE, \forall\, r \in FRE$;

These seven steps come under the family of fuzzy regular expression and star operation has a higher priority than concatenation and addition.

The language of fuzzy regular expression is defined as $L(r)$.

i. $L(\emptyset)(w) = 0, \forall\, w \in S^*$;
ii. For $r \in S \cup \{\varepsilon\}$, $L(r) = f_r$.

Where f_r is the characteristic function of fuzzy regular expression r that is explained as:

$$f_r(w) = \begin{cases} 1 & if\ w = r, \\ 0 & else; \end{cases}$$

iii. $L(\lambda r) = \lambda \otimes L(r)\ \forall\, \lambda \in \mathrm{L}$ and $r \in FRE$; where $\mathrm{L} \to [0, 1]$.
iv. $L(r_1 + r_2) = L(r_1) \vee (r_2), \forall\, r_1, r_2 \in FRE$;
v. $L(r_1 r_2) = \mathrm{L}(r_1)\, L(r_2), \forall\, r_1, r_2 \in FRE$;
vi. $\mathrm{L}(r^*) = L(r)^*\ \forall\, r_1, r_2 \in FRE$;

These are fuzzy language of fuzzy regular expression. The fuzzy language of fuzzy automaton and fuzzy regular expression should be same.

1.8 Organization of the Thesis

Chapter 1 explains the basic definitions such as lattice – ordered monoid, finite automata, fuzzy automata, position automata, follow automata and fuzzy regular expression etc.

Chapter 2 summarizes the systematic review is carried out on various researches in the field of Fuzzy Automata.

Chapter 3 explains the gap analysis and problem statement of my thesis.

Chapter 4 explains the efficient construction of fuzzy automata using follow automata from fuzzy regular expression.

Chapter 5 analyses the comparison between the existing approaches in literature for the construction of fuzzy automata.

Chapter 6 summarizes the conclusion and future work.

Chapter 2 Literature Survey

In this chapter, a systematic review has been carried out on various researches in the field of Fuzzy Automata based on various parameters such as algebraic aspects, minimization of fuzzy automata, closure properties of fuzzy automata, application of fuzzy automata, conversion of fuzzy regular expressions to fuzzy automata and various terminologies and researches related to the Fuzzy Automata.

2.1 Algebraic Aspects

Membership values in the fuzzy automata are determined using different types of lattice such as lattice – ordered monoid, complete residuated lattice, complete lattice and many other type of lattice. Li and Pedrycz [27] suggested the closure property of nondeterministic and deterministic fuzzy automata. They proved that the nondeterministic fuzzy automata are not closed under the Complement operation where as deterministic fuzzy automata are not closed under the Kleene closure operation. Further, they proved that the nondeterministic fuzzy automata are more powerful as compared with the deterministic fuzzy automata. They used the concept of lattice – ordered monoid for constructing the fuzzy automata with membership value in lattice – ordered monoid. Ignjatovic *et al.* [17] introduced complete residuated lattice for determinization of fuzzy finite automata with membership degree. Tiwari and Sharan [43] explained the concept of lattice – ordered monoid for fuzzy finite automata with membership value. For fuzzy finite automata, they proposed some topological and algebraic concept. Lattice – ordered monoid, complete residuated lattice, complete lattice and other algebraic structures can be used in fuzzy automata. Li and Pedrycz [27] used the concept of lattice – ordered monoid. Ignjatovic *et al.* [17] used the concept of complete residuated lattice, whereas Tiwari and Sharan [43] studied some algebraic and topological concept of fuzzy finite automata with membership value in lattice – ordered monoid. In fuzzy automata, membership values play a pivotal role. Different algebraic aspects are used for determining these membership values.

2.2 Minimization of Fuzzy Automata

Malik *et al*. [33] proposed the concept of minimization of fuzzy finite automata. They described that the procedure for the construction of equivalent minimal fuzzy finite automata M_2 from M_1 using the homomorphism image property. Topencharov and Peeva [45] suggested a new approach for the problem of minimization and reduction of fuzzy finite automata. They introduced an algorithm for the reduction and minimization of fuzzy automata. Ignjatovic *et al.* [17] introduced the complete residuated lattice for determinization of the fuzzy finite automata with their membership values. Further, they proposed an efficient approach for the construction of smaller automaton using the fuzzy right congruence. Malik *et al*. [33] used a classical approach and theorem for the minimization procedure, whereas Topencharov and Peeva [45] used the algebraic approach for minimization of fuzzy automata. Using the fuzzy right congruence, we can able to reduce the size of fuzzy automata. Ciric *et*. *al*. [10] described that bi-simulation can be used in various areas of computer science. Using bi-simulation, we can reduce the number of states in a system. They defined the uniform fuzzy relation, state reduction, equivalence of automata, backward and forward simulation. Ciric *et al.* [9] explained the alternating reduction by using two approaches *i.e*. left invariant fuzzy finite equivalence and right invariant fuzzy finite equivalence. They explained efficient steps for finding the fuzzy equivalence. They explained complete residutaed lattice, factor fuzzy automaton, alternate reduction, fuzzy relation equation, and state reduction. Cheng and Mo [12] proposed minimization algorithm, mealy type of fuzzy automata, minimization, and fuzzy automata. Using equivalence relations, we can reduce the number of states. Many such approaches are proposed in the literature for the minimization of fuzzy automata.

2.3 Closure Properties

Mizumoto *et al.* [31] worked on the closure properties of fuzzy automata. They explained the fuzzy language, optimistic fuzzy automata, fuzzy transition matrix and fundamental properties of fuzzy automata. Li and Pedrycz [27] introduced the concept of closure properties for nondeterministic and deterministic fuzzy automata. They proved that the nondeterministic fuzzy automata are not comes under the Complement operation and

deterministic fuzzy automata are not comes under the Kleene closure operation. In the term of power, they proved that nondeterministic fuzzy automata are more powerful than the deterministic fuzzy automata. Li *et al.* [29] described relationship between several types of fuzzy finite automata. In relationship they include nondeterministic lattice ordered monoid fuzzy automata with ε- move, nondeterministic lattice ordered monoid fuzzy automata without ε- move, deterministic fuzzy automata using lattice ordered monoid, and lattice ordered monoid fuzzy finite state machine. They also compared the power of all the fuzzy automata. Bedregal and Figueira [6] proposed that the power fuzzy Turing machine is mare than the classical Turing machine and fuzzy Turing machine are more efficient also. Bedregal and Figueira, explained universal machine, recursively enumerable set, and fuzzy function. Research is still going on for studying various properties of fuzzy automata.

2.4 Application of Fuzzy Automata

Fuzzy automata can be used in various areas such as a learning system, system modeling, neural network, pattern matching, image processing etc. Wee and Fu [48] proposed the model of learning system using the fuzzy automata. They explained that using the concept of Zadeh's fuzzy sets into mealy finite state machine, we can formulate the class of fuzzy finite automata, and the behaviour of fuzzy automaton is similar to the deterministic fuzzy automaton. They also described that fuzzy automata and stochastic automata have many similar properties. Rigatos [38] suggested that a fuzzy automaton can be used in the system modeling. He proposed two approaches named as the syntactic analysis and fuzzy automata for the fault detection. They explained how fuzzy automata are used for the pattern matching, fault detection and system modeling. Blanco *et al.* [4] suggested that how we can infer fuzzy regular grammar using two layers neural – network architecture from fuzzy examples. They explained that the fuzzy grammatical inference, fuzzy recurrent neural network and fuzzy regular grammar. Fuzzy automata can be used in the artificial intelligence. Astrain *et al.* [1] suggested how to measure the fuzziness between the strings. They defined fuzzy edit operations and string alignments approaches for measuring the fuzziness between strings. They explained the fuzzy automata with ε - move and their relationships with fuzzy automata. The main

contribution of this paper is to measure the fuzziness between strings. Wu *et al*. [49] explained the fuzzy automata system to achieve improved image processing and target recognition. Using fuzzy finite automata system, firstly good image processing achieved and then carried out for the target recognition. The fuzzy finite automata system comprises four factors *i.e*. experiment, factor extraction, target machine and image processing. They used two features *i.e*. local and global of target image and using fuzzy automata system, target recognition accomplished. They defined image processing, target recognition, fuzzy automata and proved that results of recognizing the target is more than 94.59%. Bailador and Trivino [3] introduced temporal fuzzy automata for pattern recognition. They explained pattern recognition, fuzzy models, hidden Markov model. Pattern recognition using temporal fuzzy automata is more efficient then pattern recognition using hidden Markov model. Ciric *et al.* [11] explained four type of bi-simulation *i.e*. (backward, forward, backward – forward and forward – backward bi-simulation) and two type of simulation *i.e*. (backward, forward) in fuzzy finite automata. They explained complete residuated lattice, fuzzy relation, post – fixed point, fuzzy relation inequality, bi-simulation, simulation, and fuzzy automata. They introduced an efficient approach for finding whether there is a bi-simulation/simulation between fuzzy automata and if there exists bi-simulation/ simulation between fuzzy automata then find out the greatest bi-simulation/simulation. Research is still going on for studying various applications of fuzzy automata.

2.5 Conversion of Fuzzy Regular Expressions to Fuzzy Automata

Stamenkovic and Ciric [41] introduced the new approach for the conversion of fuzzy regular expressions to fuzzy finite automata. They used the concept of position automata for the conversion of fuzzy regular expressions to fuzzy finite automata and proposed that the other methodologies for the conversion of regular expressions to non-deterministic finite automata can be extended for the same purpose. They introduced the concept of lattice–ordered monoids, fuzzy automata, fuzzy regular expressions and applications of the fuzzy automata. Li and Pedrycz [27] suggested the conversion of fuzzy regular expressions to fuzzy automata, but there is a scope of improvement in the efficiency of the conversion. Ilie and Yu [19] suggested that a novel approach for the construction of

non-deterministic finite automata (NFA) from regular expressions known as follow automata. Initially they convert the regular expressions into smaller NFA, and then eliminate the ε - transitions. Using the follow automata, the number of states will be less than or equal to n + 1 (where n = total count of input alphabets that are presented in the regular expression), whereas in position automata number of states are always n + 1. Stamenkovic and Ciric [41] introduced an efficient approach for the conversion of fuzzy regular expressions to fuzzy automata using position automata.

2.6 Various Terminologies and Researches related to the Fuzzy Automata

Cao and Ezawa [8] described that the fuzzy automata can handle the uncertainty in any system model. The concept of fuzzy finite automata has been arising from nondeterministic finite automata. To handle non-determinism in automata, the concept of non-deterministic fuzzy automata has been introduced. Nondeterministic automata have two properties such as nondeterministic fuzzy automata with the null string and nondeterministic fuzzy automata without null string. They explained all the automata such as nondeterministic fuzzy finite automata with or without empty string and deterministic fuzzy automata are comparable in the impression that they observe the same class of fuzzy language. Santos [39] suggested the concept of max-min automata. They proposed the different classes of automata, behaviour of max-min automata, homomorphism and equivalence of max-min automata. Ignjatovic *et al.* [18] introduced myhill – Nerode theorem for the fuzzy automata and language. They explained the relationship between fuzzy language, deterministic automata and fuzzy automata. Qiu [37] suggested the characterizations of fuzzy finite automata. They explained essential relationships and some fundamental concepts in fuzzy automata with bi-fuzzy property. They explained some basic concepts, properties of fuzzy finite automata and bi-fuzzy property such as bi-fuzzy topological characterization, source and successor operators, sub-automata etc. Malik *et al.* [32] introduced two concepts for determining fuzzy languages from fuzzy automata *i.e.* max – min automaton and min – max automaton. Using max – min and min – max fuzzy language, we can able to find out the membership value of a particular string in fuzzy automata. They proposed the fuzzy pumping lemma

as a necessary and sufficient condition for max – min fuzzy language. Xing [51] described the concepts of fuzzy context – free grammars, fuzzy pushdown automata, and fuzzy context – free language. Xing described the behaviour and movement of multi-stack and one – stack fuzzy pushdown automata (FPDA) for recognizing the fuzzy languages. Tiwari and Srivastava [44] suggested the decomposition of fuzzy automata in a unique form. Tiwari and Srivastava explained the concept of source – splitting sub-automaton and decomposable sub-automaton. Krithivasav and Sharda [23] proposed the concept of fuzzy ω – automata for the fuzzy ω – languages. They explained the concept of distributed fuzzy ω – finite automata and fuzzy ω – finite automata respectively for accepting the different mode such as t – mode, * - mode etc and fuzzy ω – regular language. They proved that languages accepted by distributed fuzzy ω – finite automata and the fuzzy ω – finite automata are same.

Li [30] introduced the fuzzy Turing machine using distributive lattice with membership value that is also called lattice ordered fuzzy Turing machine. He explained deterministic fuzzy Turing machine, nondeterministic Turing machine, recursively enumerable language, universal machine and fuzzy system model. He also introduced the approach of lattice – ordered nondeterministic fuzzy polynomial time computations, lattice – ordered deterministic fuzzy polynomial time computations. Belohlavek [5] proposed the concept of deterministic fuzzy finite automata. They compared the deterministic fuzzy automata and deterministic finite automata. They explained that the power of nested system of deterministic finite automata and deterministic fuzzy automata are equal, and the power of deterministic fuzzy automata and fuzzy automata is equal. Petkovic [35] suggested the concept of homomorphism and congruence of fuzzy automata and proved the theorem of homomorphism. He described lattice of congruence of fuzzy automata and algorithm for finding the maximum congruence.

Doostfatemeh and Kremer [14] described that classical automata theory deals with discrete space, whereas fuzzy automata theory deals with continuous space. Classical automata cannot manage the fuzziness in system modeling. In order to manage fuzziness in any system, fuzzy logic is incorporated with the automata theory, and the resulted fuzzy automata can deal with fuzziness in any of the system. They proposed that the

deterministic finite automata (*DFA*) are widely used in many applications, but they have some issues such as they do not deal with fuzziness etc. In particular, they emphasized on the multi membership resolution. Benlahcen and Lamotte [7] defined fuzzy automata and recalled some functional properties and theorem of fuzzy logic that are used in fuzzy automata. They proposed the synthesis method of fuzzy automata and use numerical for the illustration of the efficiency of synthesis method. Santos [40] proposed the concept of fuzzy automata and language. They described that how fuzzy automata theory is similar to classical automata theory. They used max – product composition in fuzzy automata for generating the fuzzy language. Various researches carried out in the field of Fuzzy automata are summarized in Table 2.1.

Table 2.1: Summary of various researches in the field of fuzzy automata

Parameters	**Researchers Contribution**	**Descriptions of the Research**
Algebraic Aspects	Ignjatovic et al. [17]	Introduced the new approach to construct smaller fuzzy automaton with the help of fuzzy right congruence.
	Tiwari and Sharan [43]	Explained lattice – ordered monoid for membership value in fuzzy automata and other algebraic concepts of fuzzy automata.
	Li and Pedrycz [27]	Explained some closure property of deterministic and nondeterministic fuzzy automata. Introduced the concept of fuzzy regular expressions and lattice – ordered monoids.

	Malik et al. [33]	Introduced how to minimize fuzzy finite automata M1 to equivalent fuzzy finite automata M2.
Minimization of Fuzzy Automata	Topencharov and Peeva [45]	Explained new approach for minimization and reduction of fuzzy finite automata.
	Ignjatovic et al. [17]	Introduced the new approach to construct smaller fuzzy automaton with the help of fuzzy right congruence.
	Ciric et al. [10]	Explained the benefits of bi-simulation in fuzzy automata.
	Ciric *et al.* [9]	Explained alternating reduction using right invariant and left invariant fuzzy equivalence.
	Cheng and Mo[12]	Explained the minimization algorithm. Using equivalence relation, minimize the fuzzy automata.
Closure Properties	Mizumoto et al. [31]	Suggested some closure properties of fuzzy automata.
	Li and Pedrycz [27]	Explained some closure property of deterministic and nondeterministic fuzzy automata. Introduced the concept of fuzzy regular expressions and lattice – ordered monoids.
	Li *et al.* [29]	Explained the relationship among the several types of fuzzy automata.

	Bedregal and Figueira [6]	Explained some property of fuzzy Turing machine.
Application of Fuzzy Automata	Wee and Fu [48]	Proposed model of learning system that is one of the application of fuzzy automata.
	Rigatos [38]	Explained fault detection with the help of fuzzy automata.
	Blanco et al. [4]	Explained the concept of fuzzy regular grammar and fuzzy grammatical inference.
	Astrain et al.[1]	They introduced the fuzzy automata with ε - move and measure the fuzziness between strings.
	Wu *et al*. [49]	Introduced the fuzzy automata system to achieve improved image processing and target recognition.
	Bailador and Trivino [3]	Proposed temporal fuzzy automata for pattern recognition.
	Ciric *et al.* [11]	Explained different type of bi-simulation and simulation.
Conversion of Fuzzy Regular Expressions to Fuzzy automata	Li and Pedrycz [27]	Explained some closure property of deterministic and nondeterministic fuzzy automata. Introduced the concept of fuzzy regular expressions and lattice – ordered monoids.

	Stamenkovic and Ciric [41]	Constructed the fuzzy automata from fuzzy regular expressions using position automata.
	Ilie and Yu [19]	Introduced follow automata for the conversion of regular expression to nondeterministic finite automata.
Various Terminologies and Researches related to the Fuzzy Automata	Cao and Ezawa [8]	Proposed nondeterministic fuzzy finite automata with or without ε – move and fuzzy language recognized by deterministic and nondeterministic with and without ε – moves are same.
	Santos [39]	Introduced the concept of max-min automata and behaviour of max-min automata.
	Ignjatovic et al. [18]	Introduced the myhill–nerode theorem for fuzzy automata and language.
	Qiu[37]	Explained fuzzy finite automata with bi-fuzzy property.
	Malik et al. [32]	Introduced the concept of min – max and max – min fuzzy language for fuzzy automata.
	Xing [51]	Described the working of fuzzy pushdown automata in a single stack and multi-stack.

	Tiwari and Srivastava [44]	Obtained the decomposition of fuzzy automata.
	Krithivasav and Sharda [23]	Introduced the concept of fuzzy ω - automata and define the concept of distributed fuzzy ω – automata.
	Li [30]	Explained deterministic fuzzy Turing machine, nondeterministic fuzzy automata and recursively enumerable language.
	Belohlavek [5]	Proposed the new definition of deterministic fuzzy automata.
	Petkovic [35]	Introduced the concept and proof of homomorphism and congruence of fuzzy automata.
	Doostfatemeh and Kremer [14]	Compared fuzzy automata with classical automata theory and explain various issues of fuzzy automata.
	Benlahcen and Lamotte [7]	Suggested the synthesis method of fuzzy automata.
	Santos [40]	Proposed the concept of fuzzy language and automata.

This chapter consists of systematic review on the field of fuzzy automata based on some parameters such as algebraic structure, minimization of fuzzy automata etc.

Chapter 3 Problem Formulation

To manage the system uncertainty, the concept of fuzzy automata has been proposed. Fuzzy automata can be used in many applications such as image processing, lexical analysis, description of programming and natural language, artificial intelligence, neural network, target recognition, database, fault detection, learning system, system modeling, pattern matching and detection of irregular fire frames etc. Fuzzy automata theory works well where classical automata theory fails. Fuzzy automata are similar to mealy machine that has the capability to produce the output. This chapter consists of the brief description of the problem in the field of fuzzy automata.

3.1 Gap Analysis

The nondeterministic finite automaton has been generalized into fuzzy automaton. Fuzzy automata theory works well where classical automata theory fails. Unlike the classical automata theory, fuzzy automata theory can solve many computer science problems such description of natural language, neural network etc. Classical automata theory can not deal with uncertainty. The benefit of fuzzy automata over the classical automata theory is fuzzy automata deals with uncertainty.

3.2 Problem Statement

A fuzzy automaton is a generalized version of nondeterministic finite automaton. In the field of fuzzy automata, researchers are still working on many problems. The main objectives of this thesis are explained below:

i. To analyze various techniques used in the conversion of Fuzzy Regular expression to Fuzzy Automata.
ii. Comparison of various approaches which are used for the conversion of fuzzy regular expression to fuzzy automata.
iii. Conversion of fuzzy regular expression to fuzzy automata using follow automata.
iv. To determine the fuzzy language using the concept of min – max automata.

Chapter 4 Novel Approach for the Conversion of Fuzzy Regular Expressions to Fuzzy Automata

In this chapter, the conversion of fuzzy regular expressions to fuzzy automata has been explained. In classical automata theory, numbers of methods are available for construction of finite automata from regular expressions such as Thompson's construction [46], McNaughton and Yamada and Glushkov's [15] Position automaton, partial derivative automaton of Antimirov [2] and follow automaton of Ilie and Yu [19].

Using Thompson's automata, nondeterministic finite automata with null moves from regular expressions can be constructed. In Thompson automata numbers of states are always more than n + 1, where n is the number of occurrence symbols in regular expression.

Using Glushkov's position automata, nondeterministic finite automata without null move from regular expressions can be constructed. In the position automata numbers of states are always n + 1, where n is the number of occurrence symbols in regular expression.

Follow automata and partial derivative automata are the quotient of position automata. Using follow automata, nondeterministic finite automata from regular expressions can be made but in follow automata numbers of states are less than or equal to n + 1, where n is the number of occurrence symbols in regular expression.

4.1 Conversion of Fuzzy Regular Expressions to Fuzzy automata

Let r be a random fuzzy regular expression over an input alphabet S and let L be an integral lattice – ordered monoid. Convert fuzzy regular expression (r) to regular expression ($R.E$). Consider a regular expression R.E over $S \cup Z$, where S be an input alphabet and Z are new letters that are assigned different scalar which are appearing in fuzzy regular expression [41].

Firstly follow automata M_F $(R.E) = (U^f, S \cup Z, \gamma^f, 0, B^f)$ of simple regular expression ($R.E$) is considered. After follow automata, the fuzzy automaton associated with M_{FA}

(R.E) is constructed. With the help of example, using follow automata the efficient fuzzy automaton can be designed. The advantage of using follow automata over position automata is lesser number of states without increasing the time complexity. Follow automata is the quotient of position automata.

The flow diagram showing the steps for converting fuzzy regular expressions to fuzzy automata is explained in following way:

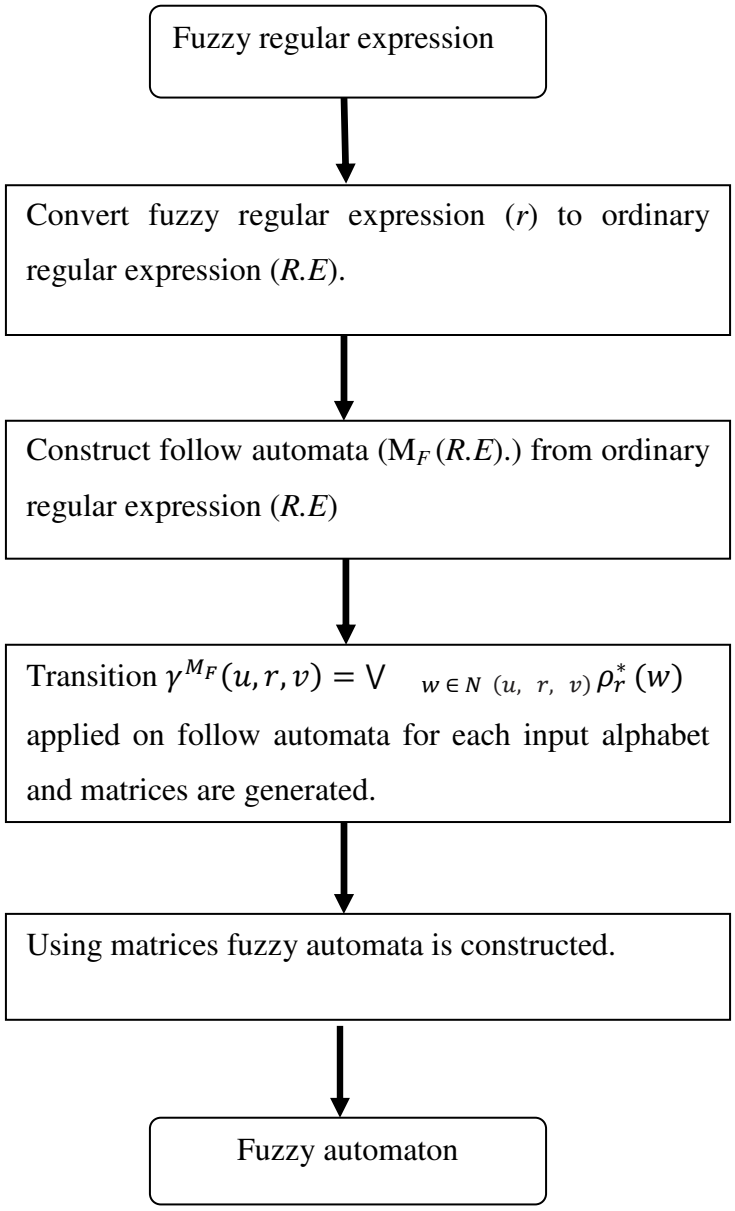


Figure 4.1: Flow graph for converting fuzzy regular expressions to fuzzy automata

Table 4.1: Comparison among several types of automata

Attribute	Number of States(|U|)	Time Complexity
Thompson automata [46]	$\|U_T\| > n + 1$	O(n)
Position automata [15]	$\|U_P\| = n + 1$	O(mn)
Partial derivative automata [2]	$\|U_F\| \leq \|U_{PD}\| \leq n + 1$	O(mn + m log m)
Follow automata [19]	$\|U_F\| \leq n + 1$	O(mn)

Example 4.1 Consider the fuzzy regular expression $r = (0.1\ a^*)\ (ba + 0.8b)^*$ over the input alphabet $S = \{a, b\}$, then convert this fuzzy regular expression to simple regular expression $(R.E) = (\alpha a^*)\ (ba + \beta b)^*$ over the input alphabet $S \cup Z = \{a, b, \alpha, \beta\}$, in this regular expression a, b comes under the input alphabet S and α, β are scalar that comes under the alphabet Z. The homomorphism mapping $\rho_r$of regular expression is explained in following way:

$$\rho_r = \begin{pmatrix} a & b & \alpha & \beta \\ 1 & 1 & 0.1 & 0.8 \end{pmatrix}$$

The marked version of regular expression R.E is $\overline{R.E} = (\alpha_1 a_2{}^*)\ (b_3 a_4 + \beta_5 b_6)^*$ and from this regular expression we can construct the follow automata.

The first, last and follow mapping is defined in following way:

First = {1},

Last = {1, 2, 4, 6},

Follow (RE, 1) = {2, 3, 5},

Follow (RE, 2) = {2, 3, 5},

Follow (RE, 3) = {4},

Follow (RE, 4) = {3, 5},

Follow (RE, 5) = {6}.

Follow (RE, 6) = {3, 5}

Using first, last and follow that is defined above, the follow automata of regular expression $\overline{R.E} = (\alpha_1 a_2^*)\ (b_3 a_4 + \beta_5 b_6)^*$ can be constructed. Follow automata M_F ($R.E$) is shown in figure 4.1. Double circle are final states and the arrow pointed to a circle is starting state.

For any random a, $b \in S$ (where S is input alphabet) and u, $v \in U_f$ (where U_f is the set of states) then,

Let us define two sets Q (u, a, v) and N (u, a, v) [41].

$$Q\ (u, a, v) = \{w \in M_z(a) \mid \gamma^{M_f}(u, a, v) = 1\}, \tag{1}$$

$$N\ (u, a, v) = \{w \in\ Q\ (u, a, v) \mid w \text{ is minimal in } Q\ (u, a, v) \text{ with respect to } \leq_{em}\ \}. \tag{2}$$

By using these two sets, we can find the minimal path and it is clear that if N (u, a, v) is $\emptyset$ then $\gamma^{M_F}(u, a, v) = 0$.

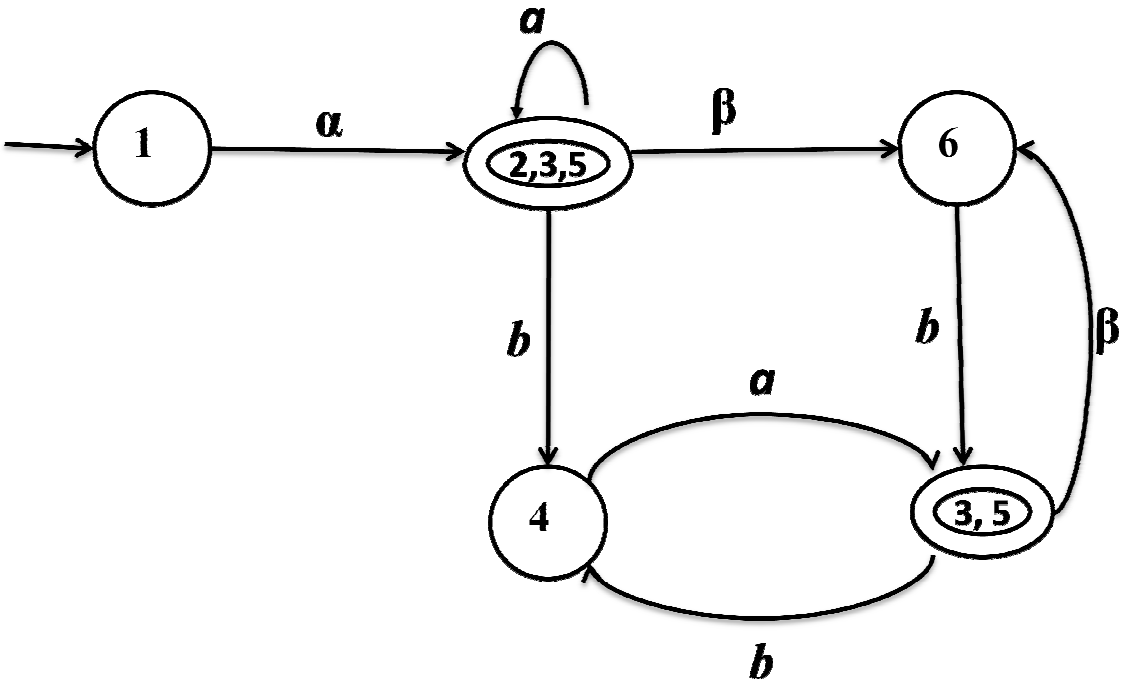


Figure 4.2: Follow automata M_F ($R.E$) for $\overline{R.E} = (\alpha_1 a_2^*)\ (b_3 a_4 + \beta_5 b_6)^*$.

It can be observed from figure 4.2, number of states in follow automata for regular expression $(\alpha_1 a_2{}^*)\ (b_3 a_4 + \beta_5 b_6)^*$ are 5 whereas the number of occurrence symbols in regular expression is 6. The final construction of figure 4.2 is defined below.

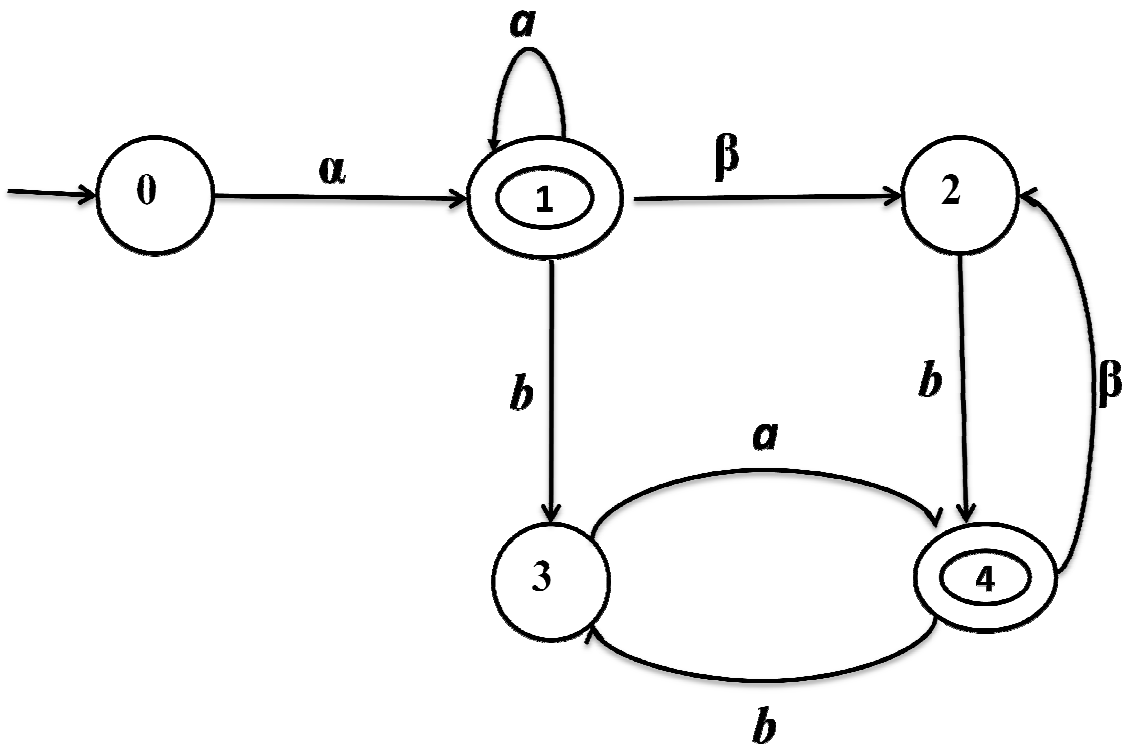


Figure 4.3: Final follow automata M_F $(R.E)$ for $\overline{R.E} = (\alpha_1 a_2{}^*)\ (b_3 a_4 + \beta_5 b_6)^*$.

Then transition [41]

$$\gamma^{M_f}(u, a, v) = \bigvee_{w \in N\ (u,\ a,\ v)} \rho_r^*(w) \qquad (3)$$

With the help of these equations, we can find out the minimum value in the path with respect to input alphabet *a*, *b*.

For input alphabet, *b*, we will find out the minimum path, and if $\gamma^{M_f}(u, b, v) = 0$, that means there is no input alphabet in that path.

$\gamma^{M_f}(0, b, 2) = \rho_r^*(\alpha\beta b\beta) = 0.1 \otimes 0.8 \otimes 1 \otimes 0.8 = 0.1,$

$\gamma^{M_f}(0, b, 3) = 0.1,$

$\gamma^{M_f}(0, b, 4) = 0.1,$

$\gamma^{M_f}(1, b, 2) = 0.8,$

$\gamma^{M_f}(1, b, 3) = 1,$

$\gamma^{M_f}(1\ b, 4) = 0.8,$

$\gamma^{M_f}(2, b, 2) = 0.8,$

$\gamma^{M_f}(2, b, 4) = 1,$

$\gamma^{M_f}(4, b, 2) = 0.8,$

$\gamma^{M_f}(4, b, 3) = 1,$

$\gamma^{M_f}(4, b, 4) = 0.8,$

$\gamma^{M_f}(u, b, v) = 0$, and

For $(u, v) \notin \{(0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (4, 2), (4, 3), (4, 4)\}$

If $\gamma^{M_f}(u, b, v) = 0$, that means there is no input alphabet *b* in that path.

For input alphabet *a*, we will find out the minimum path and if $\gamma^{M_F}(u, b, v) = 0$, that means there is no input alphabet in that path.

$\gamma^{M_f}(0, a, 1) = \rho_r^*(\alpha a) = 0.1 \otimes 1 = 0.1,$

$\gamma^{M_f}(0, a, 2) = 0.1$

$\gamma^{M_f}(1, a, 1) = 1,$

$\gamma^{M_f}(1, a, 2) = 0.8,$

$\gamma^{M_f}(3, a, 2) = 0.8,$

$\gamma^{M_f}(3, a, 4) = 1$, and

For $(u, v) \notin \{(0, 1), (0, 2), (1, 1), (1, 2), (3, 2), (3, 4)\}$

For input alphabet *b*, with the help of figure 4.3, we can observe,

$N(0, b, 2) = \{\alpha\beta b\beta\},$

$N(0, b, 3) = \{\alpha b\},$

$N(0, b, 4) = \{\alpha\beta b\},$

$N(1, b, 2) = \{\beta b\beta\},$

$N(1, b, 3) = \{b\},$

$N(1\ b, 4) = \{\beta b\},$

$N(2, b, 2) = \{b\beta\},$

$N(2, b, 4) = \{b\beta b\}$,

$N(4, b, 2) = \{\beta b\beta\}$,

$N(4, b, 3) = \{b\}$,

$N(4, b, 4) = \{\beta b\}$, and

$N(u, a, v) = \emptyset$ in all other case.

For input alphabet a, with the help of figure 4.3, it can be observed,

$N(0, a, 1) = \{\alpha a\}$,

$N(0, a, 2) = \{\alpha a\beta\}$

$N(1, a, 1) = \{a\}$,

$N(1, a, 2) = \{a\beta\}$,

$N(3, a, 2) = \{a\beta\}$,

$N(3, a, 4) = \{a\}$, and

$N(u, a, v) = \emptyset$ in all other case.

The matrices of the fuzzy transition relations $\gamma_a^{M_F}$, $\gamma_b^{M_F}$ for each input alphabet can be made and the fuzzy set of the final state B^{M_F} of fuzzy automaton $M_{FA}(r)$ are explained in following way:

For input alphabet a, fuzzy transition function is explained below:

$$\gamma_a^{M_F} = \begin{bmatrix} 0 & 0.1 & 0.1 & 0 & 0 \\ 0 & 1 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

For input alphabet b, fuzzy transition function is defined below:

$$\gamma_b^{M_F} = \begin{bmatrix} 0 & 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0.8 & 1 & 0.8 \\ 0 & 0 & 0.8 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 1 & 0.8 \end{bmatrix},$$

Fuzzy set of final state are defined in following way:

$$B^{M_F} = \begin{bmatrix} 0.1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

With the help of these matrices, we will construct the fuzzy automaton M_{FA} (r) that is explained below:

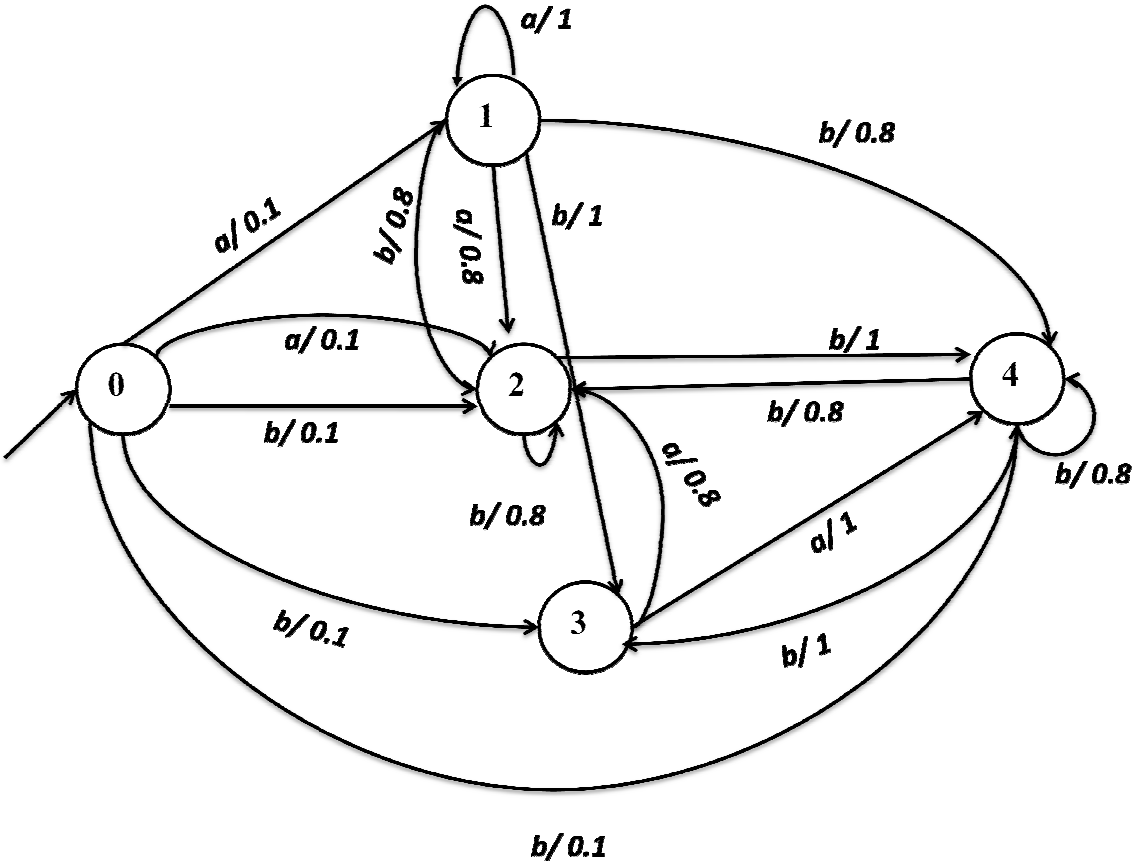


Figure 4.4: Fuzzy automata M_{FA} (r) for (0.1 a^*) $(ba + 0.8b)^*$.

Above graph is the efficient Fuzzy finite automaton of the fuzzy regular expression r = (0.1 a^*) $(ba + 0.8b)^*$ over the input alphabet S.

In this chapter, the efficient fuzzy automata are constructed using the concept of follow automata from fuzzy regular expressions.

Chapter 5 Experimental Results

In this chapter, the comparison between construction of fuzzy automata using position automata from fuzzy regular expressions and construction of fuzzy automata using follow automata from fuzzy regular expressions is done.

5.1 Comparison with Existing Approach

Example 5.1 Consider the fuzzy regular expression $r = 0.2((0.1(ab)^*)^* + b)$ over the input alphabet $S = \{a, b\}$, then convert this fuzzy regular expression to simple regular expression $(R.E) = \alpha\ ((\beta(ab)^*)^* + b)$ over the input alphabet $S \cup Z = \{a, b, \alpha, \beta\}$, in this regular expression a, b comes under the input alphabet S and α, β are scalar that comes under the alphabet Z. The homomorphism mapping ρ_r of regular expression is explained in following way:

$$\rho_r = \begin{pmatrix} a & b & \alpha & \beta \\ 1 & 1 & 0.2 & 0.1 \end{pmatrix}$$

Fuzzy regular expression $r = 0.2((0.1(ab)^*)^* + b)$ over the input alphabet $\{a, b\}$,

Regular expression $(R.E) = \alpha\ ((\beta(ab)^*)^* + b)$ over the input alphabet $\{a, b, \alpha, \beta\}$

The marked version of regular expression R.E is $\overline{R.E} = \alpha_1((\beta_2(a_3b_4)^*)^* + b_5)$ and from this regular expression we can construct the position automata.

The first, last and follow mapping is defined in following way:

First = {1},

Last = {1, 2, 4, 5},

Follow (*RE*, 1) = {2, 5},

Follow (*RE*, 2) = {2, 3},

Follow (*RE*, 3) = {4},

Follow (*RE*, 4) = {2, 3},

Follow (*RE*, 5) = {#}.

5.1.1 Fuzzy Automata using Position Automata

Using first, last and follow that is defined above, we can construct the follow automata of regular expression $\overline{R.E} = \alpha_1((\beta_2(a_3b_4)^*)^* + b_5)$.

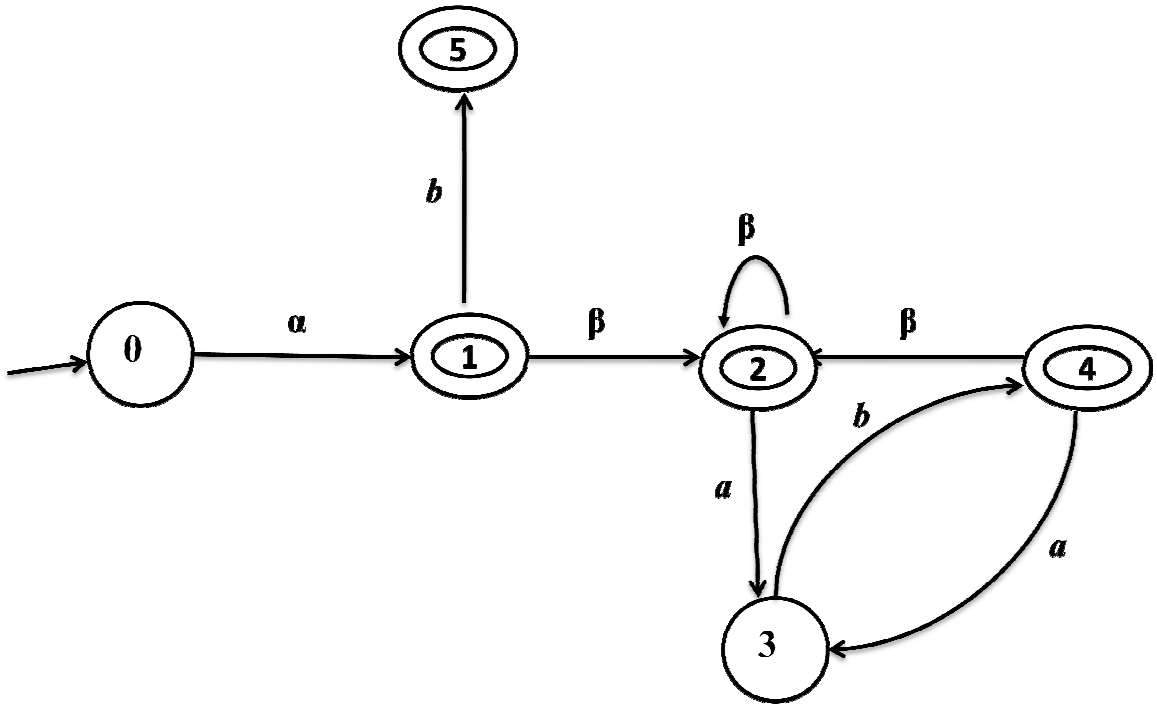


Figure 5.1: Position automata $M_P(R.E)$ for $\overline{R.E} = \alpha_1((\beta_2(a_3b_4)^*)^* + b_5)$.

Using first, last and follow that is defined above, the follow automata of regular expression $\overline{R.E} = \alpha_1((\beta_2(a_3b_4)^*)^* + b_5)$ can be constructed.

Figure 5.1 shows the position automata $M_P(R.E)$. Double circle are final states and arrow pointed a circle called initial state.

For any random $a, b \in S$ (where S is input alphabet) and $u, v \in U_f$ (where U_f is the set of states) then,

Let us define two sets $Q(u, a, v)$ and $N(u, a, v)$ [41].

$$Q(u, a, v) = \{w \in M_z(a) \mid \gamma^{M_F}(u, a, v) = 1\}, \quad (1)$$

$$N(u, a, v) = \{w \in Q(u, a, v) \mid w \text{ is minimal in } Q(u, a, v) \text{ with respect to } \leq_{em}\}. \quad (2)$$

By using these two sets, the minimal path can be found and it is clear that if $N(u, a, v)$ is $\emptyset$ then $\gamma^{M_F}(u, a, v) = 0$.

Then transition [41],

$$\gamma^{M_F}(u, a, v) = \bigvee_{w \in N\,(u,\ a,\ v)} \rho_r^*(w) \tag{3}$$

With the help of these equations, the minimum value in the path can be found with respect to input alphabet *a*, *b* and construct the fuzzy automata using position automata.

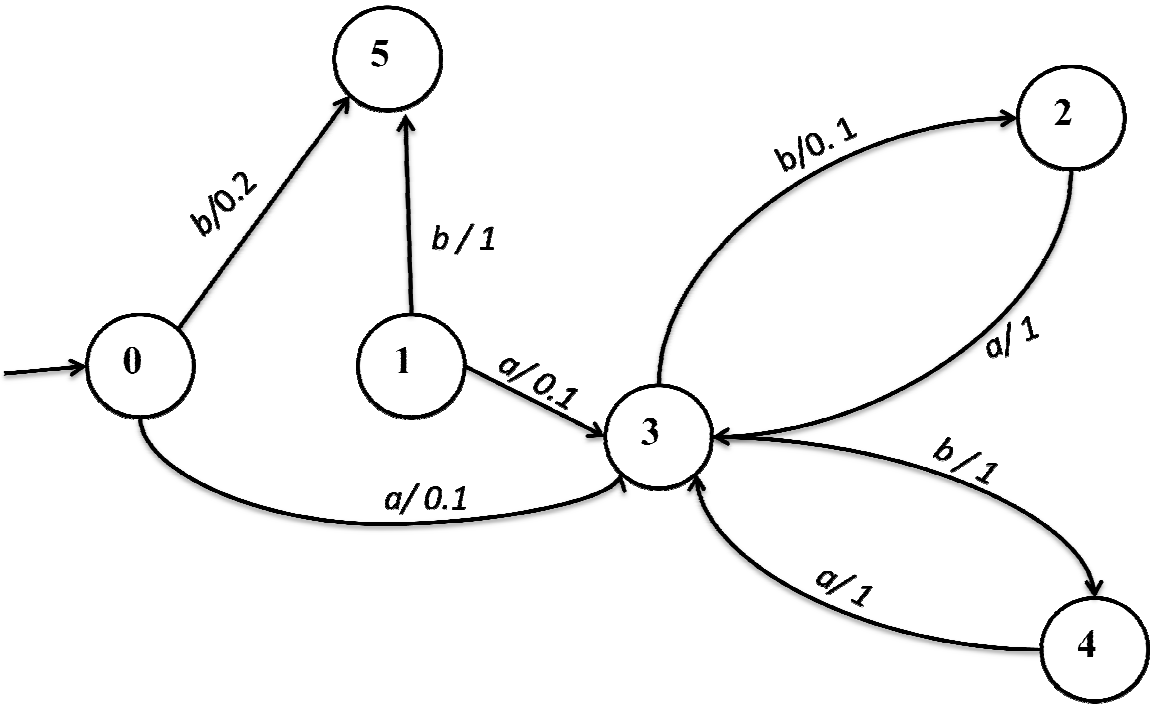


Figure 5.2: Fuzzy automata $M_{FA}(r)$ using position automata

It can be observed that construction of fuzzy automata using position automata numbers of states are 6 whereas the number of occurrence symbols in regular expression is 5. In position automata numbers of states are always n + 1.

5.1.2 Fuzzy Automata using Follow Automata

Using first, last and follow that is defined above, we can construct the follow automata of given regular expression $\overline{R.E} = \alpha_1((\beta_2(a_3b_4)^*)^* + b_5)$ and with the help of Ilie and yu's follow automata, fuzzy automata can be designed. Fuzzy automata using follow automata is explained in following way:

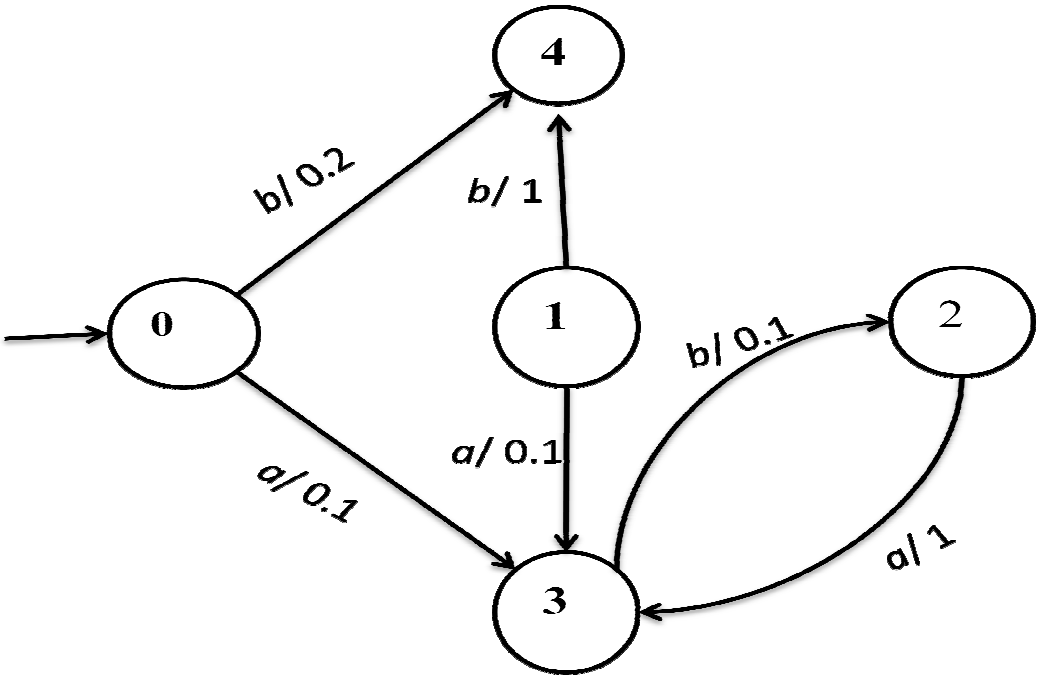


Figure 5.3: Fuzzy automata $M_{FA}(r)$ using follow automata

It can be observed that in the construction of fuzzy automata using follow automata numbers of states are always less than or equal to n + 1. Conversion of fuzzy regular expression to fuzzy automata using follow automata and position automata, both have same time complexity but the difference in number of states.

Table 5.1: Comparison between position automata and follow automata

Attribute	**Position automata [15]**	**Follow automata [19]**
Number of States $\|U\|$	$\|U_p\| = n + 1$	$\|U_F\| \leq \|U_p\|$
Complexity	O(mn)	O(mn)

It can be observed that in fuzzy automata using follow automata number of states are less than or equal to n + 1. The concept of follow automata is used for the construction of efficient fuzzy automata from fuzzy regular expressions. If follow automata from regular expressions is constructed then number of states always ≤ n + 1 (where n = total count of

input alphabets that are present in the regular expression). If regular expressions contain only addition operators (no concatenation operators) then the number of states in follow automata always 2, whereas, in position automata number of states still n + 1. If regular expressions contain only concatenation operators (no addition operators) then the number of states in both follow and position automata exactly n + 1. If regular expressions contain both addition and concatenation operators then number of states in follow automata always less than n + 1.

Theorem 5.1 Let r be a random fuzzy regular expression over the input alphabet $S \cup Z$, let L be an integral lattice – ordered monoid and L(r) recognize the language, let M_{FA} be fuzzy automata and recognize the language L(M_{FA}), then,

$$L(M_{FA}) = L(r).$$

Proof First for ε - transition, we have that

$$L(M_{FA})(\varepsilon) = B^U(u_0) = B^U(u_0) = L(r)(\varepsilon).$$

Then, for every string $w \in S^+$, where $w = v_1v_2\ldots\ldots..v_n$, and all $v_1, v_2, \ldots\ldots, v_n \in S$,

$$L(r)(w) = L(M_{FA})(w) = (\gamma_{v_0}^U \circ \ldots\ldots\circ \gamma_{v_n}^U \circ B^U)(u_0),$$

$$= \bigvee_{u_0 \ldots u_n \in U} (\gamma_{v_1}^U)(u_0, u_1) \otimes \ldots\ldots \otimes (\gamma_{v_n}^U)(v_{n-1}, v_n) \otimes (B^U)(u_0)$$

$$= (\gamma_{v_1}^U \circ \ldots\ldots.. \circ \gamma_{v_n}^U \circ B^U)(u_0)$$

$$= L(M_{FA}).$$

In this chapter, Comparison between the construction of fuzzy automata using position automata and the construction of fuzzy automata using the approach of follow automata has been performed. The construction Fuzzy automata using follow automata have lesser number of states as compared to position automata. With the help of the theorem, it has been proved that language accepted by the fuzzy regular expression and fuzzy automata are same.

Chapter 6 Conclusion and Future Scope

In this chapter, conclusion of this thesis and future scope of fuzzy automata is discussed.

6.1 Conclusion

Classical automata theory can not deal with uncertainty. To deal with system uncertainty, nondeterministic finite automata have been generalized into fuzzy automata. Stamenkovic and Ciric proposed an approach using the position automata for the construction of fuzzy automata from fuzzy regular expressions. In this thesis, the concept of follow automata has been extended for the metamorphosis of fuzzy regular expressions to fuzzy automata. Follow automata is a quotient of position automata. Compared with the existing approaches in the literature, the proposed approach will generate a lesser number of states without increasing the time complexity.

6.2 Future Scope

Fuzzy automata can be used in various applications and there is a scope of future research work can be carried out in the field of Fuzzy automata. Future work will concentrate on the following areas:

i. Design of an efficient approach for the conversion of fuzzy automata to fuzzy regular expressions.
ii. Pumping lemma for fuzzy language.
iii. Max product automata can be used for finding fuzzy language.
iv. More application of fuzzy automata can be explored such as target recognition, image processing etc.

References

[1] Astrain, J. J., Mendívil, J. R. G. and Garitagoitia, J. R. (2006) "Fuzzy automata with ε-moves compute fuzzy measures between strings", *Fuzzy Sets and Systems*, vol. 157, pp. 1550 – 1559.

[2] Antimirov, V. (1996) "Partial derivatives of regular expressions and finite automaton constructions", *Theoretical Computer Science*, Vol. 155, pp. 291–319.

[3] Bailador, G. and Trivino, G. (2010) "Pattern Recognition Using Temporal Fuzzy Automata", vol. 161, pp. 37 – 55.

[4] Blanco, Delgado, M. and Pegalajar (2001) "Fuzzy automaton induction using neural network", *International Journal of Approximate Reasoning,* vol. 27, pp. 1 – 26.

[5] Belohlavel, R. (2002) "Determinism and fuzzy automata", *Information Sciences*, vol. 143, pp. 205 – 209.

[6] Bedregal, B. C. and Figueira, S. (2008) "On the Computing Power of fuzzy Turing machine", *Fuzzy Set and System*, vol. 159, pp. 1072 – 1083.

[7] Benlahcen, D. and Lamotte, M. (1981) "A Fuzzy Automaton Synthesis Method", *International Federation of Automatic Control*, vol. 17, pp. 299 – 306.

[8] Cao, Y. and Ezawa, Y. (2012) "Nondeterministic fuzzy automata", *Information Sciences*, vol. 191, pp. 86-97.

[9] Ciric, M., Stamenkovic, A., Ignjatovic, J. and Petkovic, T. (2010) "Fuzzy relation equation and reduction of fuzzy automata", *Journal of Computer and System Sciences*, vol. 69, pp. 609 – 633.

[10] Ciric, M., Ignjatovic, J., Damljanovic, N. and Basic, M. (2012) "Bisimulations for fuzzy automata," *Fuzzy Sets and Systems*, vol. 86, pp. 100 – 139.

[11] Ciric, M., Ignjatovic, J., Jancic, I. and Damljanovic, N. (2012) "Computation of the greatest simulations and bisimulations between fuzzy automata", *Fuzzy Sets and Systems*, vol. 208, pp. 22 – 42.

[12] Cheng W. and Mo Z – W. (2004) "Minimization algorithm of fuzzy finite automata," *Fuzzy Set and System*, vol. 141, pp. 439 – 448.

[13] Ciric, M., Ignjatovic, J., Jancic, I. and Damlljanavic, N. (2012) "Computation of the greatest simulation and bisimulation between fuzzy automata", *Fuzzy Set and System*, vol. 208, pp. 22 – 42.

[14] Doostfatemeh, M. and Kremer, S.C. (2005) "New directions in fuzzy automata", *International Journal of Approximate Reasoning*, vol. 38, pp. 175 – 214.

[15] Glushkov, V.M. (1961) "The abstract theory of automata", *Russ. Math. Surv.* Vol. 16, pp. 1–53.

[16] Guman, S. K. and Kumar, A. (2012) "Union-freeness of Regular Languages", *International Journal of Computer Applications*, vol. 50(4), pp. 6 – 8.

[17] Ignjatovic, J., Ciric, M. and Bogdanovic, S. (2008) "Determinization of fuzzy automata with membership values in complete residuated lattices", *Information Sciences*, vol. 178, pp. 164 – 180.

[18] Ignjatovic, J., Ciric, M., Bogdanovic, S. and Petkovic, T. (2010) "Myhill–Nerode type theory for fuzzy languages and automata", *Fuzzy Sets and Systems*, vol. 161, pp. 1288 – 1324.

[19] Ilie, L. and Yu, S. (2003) "Follow automata", *Information and computation,* vol. 186, pp. 140-162.

[20] Kumar, A. and Verma, A. K. "Conversion of Parallel Regular Expressions to Non-deterministic Finite automata using partial derivatives", *Chiang Mai Journal of Science*, accepted for publication.

[21] Kumar, A. and Verma, A. K. (2014) "A Novel Algorithm for the Conversion of Parallel Regular Expressions to Non-deterministic Finite automata", *Applied mathematics & Information Science*, vol. 8(1), pp. 95 – 105.

[22] Kumar, A. (2013) "Design and Development of algorithms for the Conversion of Parallel Regular Expressions to Deterministic Finite Automata", Ph. D. Thesis, Thapar University.

[23] Krithivasan, K. and Sharda, K. (2001) "Fuzzy ω - automata", *Information Science*, vol. 138, pp. 257 – 281.

[24] Lee, E.T. and Zadeh, L.A. (1969) "Note on fuzzy languages", *Information Sciences*, vol. 1, pp. 421- 434.

[25] Linz, P. (2000) "An Introduction to Formal Languages and Automata", 3rd

edition, Jones and Bartlet Publishers.

[26] Li, Y. and Pedrycz, W. (2004) "Regular expressions with truth values in lattice-monoid and their languages", Proc NAFIPS '04, vol. 02, pp. 572 – 577.

[27] Li, Y. M. and Pedrycz, W. (2005) "Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids", *Fuzzy Sets System*, vol.156, pp. 68–92.

[28] Li, Y. and Liang, C. (2008) "Algebraic properties on the cuts of lattice-valued regular languages", *Soft Computing*, vol.12, pp. 1049 – 1057.

[29] Li, Z., Li, P. and Li, Y. (2006) "The relationships among several type of fuzzy automata", *Information Sciences*, vol. 176, pp. 2208 – 2226.

[30] Li, Y. (2009) "Lattice valued fuzzy Turing machine computing power universality and efficiently", *Fuzzy Set and System*, vol. 160, pp. 3453 – 3474, 2009.

[31] Mizumoto, M., Toyoda, J. and Tanaka, K. (1969) "Some Considerations on Fuzzy Automata", *Journal of Computer and System Science*, vol. 3, pp. 409 – 422.

[32] Malik, D. S., Mordeson, J. N. and Sen, M. K. (1969) "On fuzzy regular language", *Information Science*, vol. 88, pp. 263 – 273.

[33] Malik, D. S., Mordeson, J. N. and Sen, M. K. (1999) "Minimization of fuzzy finite automata", *Information Science*, vol. 113, pp. 323 – 330.

[34] Mordeson, J. N. and Malik, D.S. (2002) "Fuzzy Automata and Languages: Theory and Applications", *Chapman & Hall*, *CRC*, *Boca Raton*, London.

[35] Petkovic, T. (2006) "Congruences and homomorphisms of fuzzy automata", *Fuzzy Sets and Systems*, vol. 157, pp. 444 – 458.

[36] Peeva, K. and Zahariev, Z. (2008) "Computing behavior of finite fuzzy machines—algorithm and its application to reduction and minimization", Information Science vol. 178, pp. 4152–4165.

[37] Qiu, D.W. (2004) "Characterizations of fuzzy finite automata", Fuzzy Sets System, vol. 141 pp. 391–414.

[38] Rigatos, G. G. (2009) "Fault detection and isolation based on fuzzy automata", *Information Science*, vol. 179, pp. 1893 – 1902.

[39] Santos, E. S. (1968) "Maximin automata", *Info. Control*, vol. 12, pp. 367-377.

[40] Santos, E. S. (1976) "Fuzzy automata and languages", *Information Sciences*, vol. 10, pp. 193–197.

[41] Stamenkovic, A. and Ciric, M. (2012) "Construction of fuzzy automata from fuzzy regular Expressions", *Fuzzy Sets and Systems,* vol. 199, pp. 1-27.

[42] Singh, K. and Loura, A. K. (2011) "Comparisons amongst Different Techniques for Conversion of Deterministic Finite Automata to Regular Expression", *International Journal of Advanced Research in Computer Science*, vol. 2(3) pp. 125 – 128.

[43] Tiwari, S. P. and Sharan, S. (2012) "Fuzzy Automata Based on Lattice-ordered Monoid with Algebraic and Topological Aspects", *Fuzzy Information Eng.* Vol. 2, pp. 155 – 164.

[44] Tiwari, S. P. and Srivastava, A. K. (2005) "On a decomposition of fuzzy automata", *Fuzzy Sets and Systems*, vol. 151, pp. 503 – 511.

[45] Topencharov, V. V. and Peeva, K. G. (1981) "Equivalence, Reduction and Minimization of Finite Fuzzy-Automata", *Journal of Mathematical Analysis and Applications*, vol. 84, pp. 270 – 281.

[46] Thompson, K. (1968) "Regular expression search algorithm", *Commun. ACM* vol. 11 (6), pp. 419–422.

[47] Ullman, J. D., Hopcroft J. E. and Motwani, R. (2001) "Introduction to automata theory language and computation", 2nd Edition, Pearson Education.

[48] Wee, W.G. and Fu, K.S. (1969) "A formulation of fuzzy automata and its application as a model of learning systems", IEEE Transactions Systems Man cybernetics, vol. 5, pp. 215–223.

[49] Wu, Q – E., Pang, X – M. and Han, Z – Y. (2011) "Fuzzy automata system with application to target recognition based on image processing", *Computer and Mathematic with application*, vol. 61, pp. 1267 – 1277.

[50] Wee, W. G. (1967) "On Generalizations of Adaptive Algorithm and Application of the Fuzzy Sets Concept to Pattern Classification", Ph.D. Thesis, Purdue University.

[51] Xing, H. (2007) "Fuzzy pushdown automata," *Fuzzy Sets and Systems*, vol. 158, pp. 1437 – 1449.

List of Publications

[1] **Singh, R. K.** and Kumar, A. "Metamorphosis of Fuzzy Regular Expressions to Fuzzy Automata using the Follow Automata," ***International Journal of Computer Mathematics (Taylor and Francis*)**.

[2] **Singh, R. K.** and Kumar, A. "A Novel approach for Determination of the Membership value of the Strings in Fuzzy Language," *Seventh International Conference on Contemporary Computing* ***IEEE*,** (IC3' 2014), Noida, India, August, 7 – 9, 2014.

[3] **Singh, R. K.** and Kumar, A. "Fuzzy Automata: Systematic Review," *International Conference on Interdisciplinary Advance in Applied Computing* ***ACM***, (ICONIAAC '14), Kerala, India, October, 10 – 11, 2014.