Some Improvements in the Construction of Fuzzy Automata

A Thesis

Submitted in fulfillment of the requirement

for the award of the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND ENGINEERING

by
Sunita Garhwal
(Registration No. 951311003)

Under the guidance of
Dr. Ram Jiwari
(Asst. Professor)
(Department of Mathematics, IIT Roorkee)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2017

DEDICATED WITH EXTREME AFFECTION AND

GRATITUDE TO

My Parents and My Daughters

Dishitta and Anaya Loura

CERTIFICATE

I hereby certify that work which is being presented in the thesis entitled *"Some Improvements in the Construction of Fuzzy Automata"* in fulfillment of the requirements of **DOCTOR OF PHILOSOPHY** submitted in the **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**, THAPAR UNIVERSITY, PATIALA is an authentic record of my own work carried out under the supervision of **Dr. Ram Jiwari** and refers work of other researchers which are duly listed in reference section.

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

Sunita

(Sunita Garhwal)

Registration No. 951311003

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

Rm

(Dr. Ram Jiwari)

Supervisor and Assistant Professor

Department of Mathematics

Indian Institute of Technology Roorkee, India

ACKNOWLEDGEMENT

First and foremost, I would like to thank Almighty God, the most gracious and merciful, for providing me this opportunity and granting me the capability to proceed successfully.

I take this opportunity to express my profound gratitude and deep regard to my respected Supervisor, *Dr. Ram Jiwari, Assistant Professor, Department of Mathematics, Indian Institute of Technology, Roorkee, India* for his invaluable guidance, support, and encouragement throughout the course of this thesis. His positive outlook and confidence in my research inspired me and gave me confidence to complete this thesis. His deep insights helped me at various stages of my research. I feel very lucky to get a lifetime opportunity to work under his knowledgeable supervision. It has been my great pleasure and honor to have worked with him.

I also express my gratitude to the Doctoral committee comprising of Dr. Maninder Singh (Head, CSED), Dr. Rinkle Rani (Associate Professor, CSED), Dr. Maninder Kaur, (Assistant Professor, CSED), and Dr. Amit Kumar (Associate Professor, SOM) for monitoring the progress and providing valuable suggestions for the improvement of this Ph.D. work. I am very thankful to Dean of Research and Sponsored Projects Dr. O.P. Pandey for his valuable inputs and suggestions.

I would also take this opportunity to mention my deep gratitude towards Dr. Sapna, wife of my respected Supervisor, Dr. Ram Jiwari whose presence was always there whenever I needed in both my personal and professional life. Working with both of you was my great honor and I always felt like family support from both of you. I heartily thankful to both of you for supporting during entire duration of my Ph.D.

I would like to acknowledge the most important persons of my life – my parents Birma Devi, Tarachand Garhwal and my husband Dr. Ajay Kumar for their valuable support, selfless love, wishes, care and encouragement throughout my life. I am also indebted to my Sister cum friend Rekha Garhwal not only for all her useful suggestions but also for being there to listen when I needed an ear. My brothers Mr. Sunil Garhwal and Anil Garhwal have also supported me and encouraged me to carry out this work. In spite of being my younger brothers, they always stand beside me at every point of my life.

Sincere, thanks to all my friends especially Ruchika Lamba, Geeta Kasana, Sonia, Maninder Kaur, Vineeta Bassi, Parul Batra, Shruti Sharma and Nidhi Kalra for their

support and cooperation throughout the entire process. Thanks for the friendship and memories.

I would like to thank the members of CSED Department, faculty members and staff members who were always there at need of the hour and provided with the help and facilities, which I required for the completion of my thesis.

I would like to thank my two little daughters Dishitta and Anaya Loura. They have given me so much happiness, love and keep me hoping. Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

Sunita

(Sunita Garhwal)

ABSTRACT

Regular expression is a mathematical representation of regular language. There are many recent developments in Science, Engineering, and Mathematics in the previous century. In formal languages, string is either totally accepted or rejected, whereas in case of fuzzy languages, the string is not totally accepted or rejected, rather it belongs to the language with some degree of membership. One of the main concerns related to these developments is uncertainty. For handling uncertainty and reducing the gap between natural and programming language, the concept of fuzzy languages have been introduced as an extension of formal languages. Fuzzy automata are generalization of the classical automata, closing the gap between classical automata theory and natural languages. Fuzzy regular expressions and fuzzy automata have received significant attention due to their practical applications in various fields, such as databases, pattern recognition, control systems, discrete event systems, clinical monitoring, computing with words, learning systems, and descriptions of natural and programming languages.

The main aim of this dissertation is to propose some improvements in the construction of fuzzy automata. The inclusion of shuffle operators in the regular expression is known as a parallel regular expression. Motivated by the applications of fuzzy automata and parallel regular expressions, we have proposed a new mathematical model that involves fuzziness in parallel finite automata. In this thesis, an algorithm is proposed for the conversion of parallel fuzzy regular expressions to ε-free fuzzy automata.

Motivated by the existing literature for the transformation of finite automata to regular expression using approaches such as Arden's Theorem, Transitive closure, Brzozowski algebraic and State removal, in this thesis, an algorithm is proposed for the conversion of a fuzzy automaton into a fuzzy regular expression by extending the well-known concept of the Transitive closure method. Furthermore, we have proved the equivalence of fuzzy automata and fuzzy regular expressions.

Sempere introduced the concept of linear expressions to represent the linear context-free languages. Motivated by the existing work of Sempere, we have applied the concept of linear expressions for representing Inverted Repeats, Hairpin structures, Stem and Loop structures. Further, we have proposed the concept of fuzzy linear expressions. An algorithm is proposed for the conversion of fuzzy linear grammar to fuzzy linear

expressions. At last, we have extended the concept of Antimirov's partial derivatives for the conversion of fuzzy linear expressions to fuzzy automata.

Keywords: Fuzzy regular expression, Fuzzy automata, Shuffle operator, Partial derivative, Two-sided partial derivative, Fuzzy linear expressions, Fuzzy context-free grammar.

PREFACE

This manuscript presents my research work done at the Computer Science and Engineering Department, Thapar University, Patiala, India from Jan 2014 to July 2017. This manuscript mainly focuses on proposing some improvements in the construction of fuzzy automata. The work presented in this manuscript is as follows:

1. Survey of the existing literature on fuzzy regular expressions and fuzzy automata.
2. Introduced the concept of parallel fuzzy regular expression.
3. An algorithm is proposed for the conversion of parallel fuzzy regular expression to epsilon-free fuzzy automaton.
4. An algorithm is designed for the conversion of a fuzzy automaton to a fuzzy regular expression using transitive closure method.
5. Concept of linear expression is applied to represent Bio-molecular structures named as Inverted Repeats, Hairpin structures, Stem and Loop.
6. Introduced the concept of fuzzy linear expression.
7. Proposed a new variant of Antimirov's partial derivatives for the conversion of fuzzy regular expressions to ε -free fuzzy non-deterministic finite automata.
8. Proposed an algorithm for the conversion of fuzzy linear grammar to fuzzy linear expression.

The list of the main research papers accepted and communicated that I have authored or co-authored during my Ph.D. thesis are as follows:

RESEARCH PUBLICATIONS

1. S. Garhwal and R. Jiwari, Parallel Fuzzy Regular Expression and its Conversion to Epsilon-Free Fuzzy Automaton, The Computer Journal, 59(9), 1383-1391, 2016. [SCIE Indexed, Impact Factor-0.711].

2. S. Garhwal and R. Jiwari, Conversion of fuzzy automata into fuzzy regular expressions using transitive closure, Journal of Intelligent & Fuzzy Systems, 30(6), 3123-3129, 2016 [SCIE Indexed, Impact Factor-1.261].

3. S. Garhwal, R. Jiwari, and S. Tomasiello, Revising Antimorov's partial derivatives for fuzzy regular expressions, **accepted** in International Conference on

High Performance Computing & Simulation (HPCS 2017), Genoa, Italy, July 17-21, 477-482, 2017.

4. S. Garhwal and R. Jiwari, Fuzzy linear expressions and its conversion to fuzzy automata, **will be communicated** to Soft Computing.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

RE: Regular Expression

DFA: Deterministic Finite Automata

NFA: Non-deterministic finite automata

FLAT: Formal Language and Automaton Theory

LEX: Lexical Analyzer

LIST OF FIGURES

LIST OF TABLES

Chapter-1

INTRODUCTION

This chapter aims to introduce the preliminary concepts, basic notations, motivation and organization of the thesis.

1.1 MOTIVATION

There are many recent developments in Science, Engineering and Mathematics in the previous century. One of the main concerns related to these development are uncertainty. For handling uncertainty, and reducing the gap between natural and programming language, the concept of fuzzy languages has been introduced as an extension of formal languages. In formal languages, the string is either accepted or rejected, whereas, in the case of fuzzy languages, the string is not totally accepted or rejected, rather it belongs to the language with some degree of membership. Zadeh [1] introduced the concept of fuzzy set in 1965. The concept of fuzzy languages was introduced by Santos [2]. Classical formal languages are a prime example of the computational system over discrete spaces, whereas the fuzzy automata are the computational system over continuous spaces.

In classical formal languages, various models of computation can be defined mathematically. Some of the computational models are as powerful as actual computers, whereas others are not. The simplest type of abstract model is finite automata, and the underlying principle is that the system at any moment of time is in a finite number of discrete states and moves among these states in a predictable way based upon the input signals.

Regular expressions are a mathematical representation of regular languages. Regular expressions play an important role in the field of computer science such as pattern matching, protocol conformance testing, compiler design, text processor, DNA computing [3-7].

Fuzzy automata are a generalization of classical automata [8]. In the last decade, fuzzy automata have become an invaluable tool for reducing the gap between natural languages and classical automata theory. Fuzzy regular expressions and fuzzy automata have received significant attention due to their practical applications in various fields, such as databases, pattern recognition, control systems, discrete event systems, clinical

monitoring, computing with words, learning systems, and descriptions of natural and programming languages [8-16].

Example 1.1. Consider a fuzzy language L_1 defined over $\{0,1\}$ and the membership of a string is determined by the function $f(w)$:

$$f(w)=\begin{cases}1 & w\in 0^*10^*\\ 0.7 & w\in 0^*\\ 0.7 & w\in 0^*10^*10^*\\ 0.5 & w\in 0^*10^*10^*10^*\\ 0 & otherwise\end{cases}$$

Strings $w_1=0010, w_2=001010, w_3=00101010$ and $w_4=00$ are accepted with membership values $1, 0.7, 0.5$ and 0.7 respectively.

Since 1960, the membership values in fuzzy automata and languages have been considered from the Gödel structure [14]. In recent years, the membership values in fuzzy automata have derived from certain algebraic structures such as the lattice-ordered monoid [17-22] and the complete residuated lattice [23-24]. A lattice is a poset in which every pair should have distinct greatest lower bound (infimum) and least upper bound (supermum) [25]. Kakoty et al. [26] proposed the fuzzy logic model to evaluate expertise level of learner in an e-learning environment. Okamato et al. [27-28] designed a rule-based inference method and evaluate their effect in natural language proposition involving fuzzy quantifiers.

Stamenkovic and Ćirić [17] developed an approach for the conversion of fuzzy regular expressions to fuzzy automata using the concept of position automata [29-30]. They treated the scalar appearing in the fuzzy regular expression as the extended alphabet. By applying the concept of position automata, they have converted fuzzy regular expressions into fuzzy automata. Li and Pedrycz [21] considered the integral lattice-ordered monoid and proved that a fuzzy language can be represented by a fuzzy regular expression if, and only if, a fuzzy automaton can represent it. Li et al. [22] introduced the concept of ε-move in fuzzy automata. Kuske [31] proposed a similar approach for the weighted regular expression over a semiring. Mendivil and Garitagotia [18] extended the Stamenkovic and Ćirić [17] approach and removed the ε-transition from the fuzzy automata.

The inclusion of shuffle operators in the regular expression is known as a parallel regular expression. It appears in various fields of computer science such as concurrency of processes, process algebra, multipoint communication, plan recognition, XML schema language Relax NG, and modeling and verification of systems [6, 32-36].

Estrade et al. [32-33] proposed an approach to obtain parallel finite automata from regular expressions by including shuffle operator. Kumar and Verma [37] proposed a novel approach for obtaining non-deterministic finite automata from extended regular expression by using the shuffle operator. Further, they [38] proposed a novel approach for obtaining ε-free non-deterministic finite automata from parallel regular expressions. Stotts and Pugh [7] introduced a composition rule for obtaining parallel finite automata using interleaving operator. Gelade [39] proposed an approach for the conversion of extended regular expressions using interleaving, intersection and counting operator to regular expressions, deterministic finite automata and non-deterministic finite automata.
In the literature, there exist many approaches for the transformation of the finite automaton to a regular expression such as Arden's Theorem [40], Transitive closure [41], Brzozowski algebraic [42], and State elimination [43]. Neumann [44] compared three approaches namely: Transitive closure, Brzozowski algebraic and state elimination for the transformation of finite automata to a regular expression. Transitive closure is simple and easy to implement. State elimination is difficult to implement but easy to recognize the regular expression manually. Brzozowski algebraic is easy to implement but best-suited for recursive languages.

Yamamoto [45-46] proposed an approach for the conversion of semi-extended regular expressions to non-deterministic finite automata (NFA). Thomason and Marinos [47] proposed a method for constructing deterministic fuzzy finite automata and its corresponding unambiguous grammar for the fuzzy regular expression. Cao and Ezawa [48] proposed the concept of nondeterministic fuzzy finite automata and recognized the corresponding fuzzy language.

Partial derivatives are used in classical automata theory (for details, refer to [42, 49-56]) for the conversion of regular expressions to finite automata [56], tree regular expressions to tree automata [50], and ω-regular expressions to Büchi automata [49]. Use of Antimirov's partial derivatives [56] in classical automata theory is an important issue

because it generates a smaller ε-free non-deterministic finite automaton with a maximum of $(m+1)$ states, where m is the number of instance of symbol in the regular expression.

1.2 PRELIMINARIES CONCEPT

In this section, we introduce some of the basic notations and definitions which are used throughout the thesis.

Let Σ denotes an alphabet that consist of symbols/letters, ε represents the empty string. Σ^* [57] represents the free monoid generated by concatenation on Σ, including the empty string. The length of a word w is denoted by $|w|$. A language over Σ is a subset of Σ^*. Every word $w \in \Sigma^*$ has a degree of membership value in f given by $f(x) \in L$ [21].

DEFINITION 1.2.1. A regular expression (RE) [57] can be recursively defined as follows:

- ε is a regular expression representing regular language $\{\varepsilon\}$,
- ϕ is a regular expression representing regular language $\{\}$,
- a is a regular expression representing regular language $\{a\}$,
- If r_1 and r_2 are regular expressions representing regular languages L_1 and L_2, then

  r_1^* represent a regular expression representing the Kleene closure of L_1.

  $r_1 r_2$ represent a regular expression representing the concatenation of L_1 with L_2.

  $r_1 + r_2$ represent a regular expression representing the union of L_1 and L_2.

All regular expressions can be obtained by applying above rules finite number of times.

Example 1.1. Regular expression for the language over $\Sigma = \{0,1\}$ that accepts all the strings having different first and last symbol is as follows:

Consider, language L_1 consists of strings that start with 0 and ends with 1 is $0(0+1)^*1$ and language L_2 consists of strings that start with 1 and ends with 0 is $1(0+1)^*0$.

The final regular expression is $0(0+1)^*1+1(0+1)^*0$.

DEFINITION 1.2.2. Grammar [57] is a quadruple $G(N, T, P, S)$, where

- N is a finite set of non-terminals,
- T is a finite set of terminals,
- P is a set of productions of the form $\alpha \to \beta$ where $\alpha, \beta \in (N \bigcup T)^*$, and

- S is a start symbol, and $S \in N$.

DEFINITION 1.2.3. Grammar is said to be regular [40] if it is either left linear or right linear. A right linear grammar consists of production rules of the form $A \to \alpha_1 B \mid \alpha_2$ where $A, B \in N$ and $\alpha_1, \alpha_2 \in T^*$. A left linear grammar consists of production rules of the form $A \to B\alpha_1 \mid \alpha_2$ where $A, B \in N$ and $\alpha_1, \alpha_2 \in T^*$.

Example 1.2. Right and left linear grammar for a regular expression $0(0+1)^*1+1(0+1)^*0$ is as follows:

Right linear grammar:

$S \to 0A \mid 1B$

$A \to 0A \mid 1A \mid 1$

$B \to 0B \mid 1B \mid 0$

Given string $w = 0111$

$S \to 0A \to 01A \to 011A \to 0111$

Left linear grammar:

$S \to A1 \mid B0$

$A \to A0 \mid A1 \mid 0$

$B \to B0 \mid B1 \mid 1$

Given string $w = 0111$

$S \to A1 \to A11 \to A111 \to 0111$

DEFINITION 1.2.4. A deterministic finite automaton (DFA) [40] is a quintuple $N = (Q, \delta, \Sigma, F, q_0)$ where

- Q represents a finite set of states,
- δ is a transition function mapping from $Q \times \Sigma \to Q$,
- Σ is an alphabet,
- $q_0 \in Q$ is start state, and
- $F \subseteq Q$ denotes final states.

Fig. 1.1 represents the transition diagram for the regular expression $0(0+1)^*1+1(0+1)^*0$.

DEFINITION 1.2.5. A non-deterministic finite automaton (NFA) [40] is a quintuple $N = (Q, \delta, \Sigma, F, q_0)$ where

- Q represents the finite set of states,
- δ is a transition function mapping from $Q \times \Sigma \to 2^Q$,
- Σ is an alphabet,
- $q_0 \in Q$ is a start state, and
- $F \subseteq Q$ denotes final states.

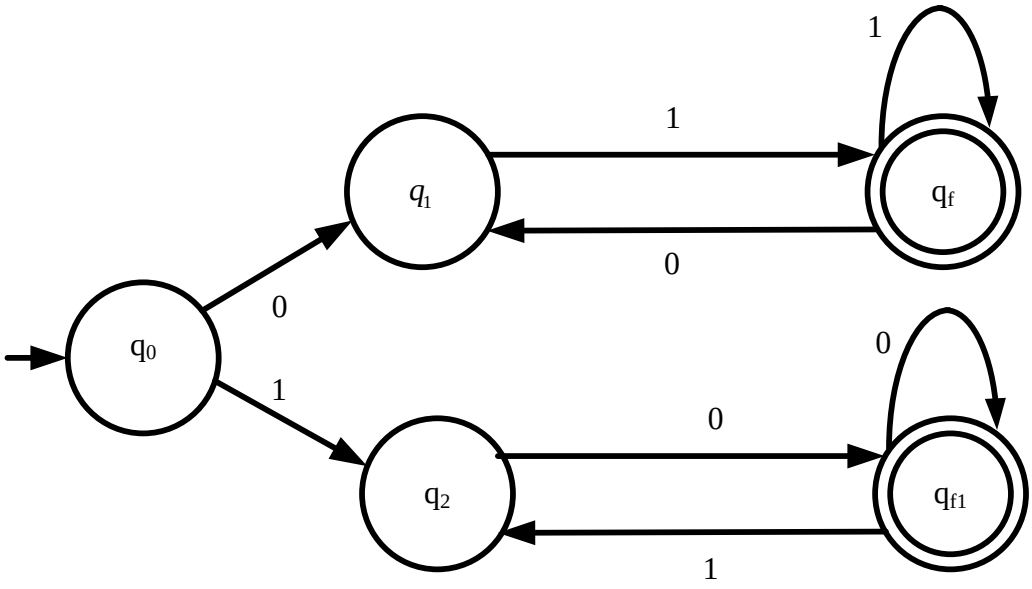

Fig. 1.1: DFA for a regular expression $0\,(0+1)^*1+1\,(0+1)^*0$.

Example 1.3. Non-deterministic finite automaton for the language over $\Sigma = \{0,1\}$ that accepts all strings that end with 11.

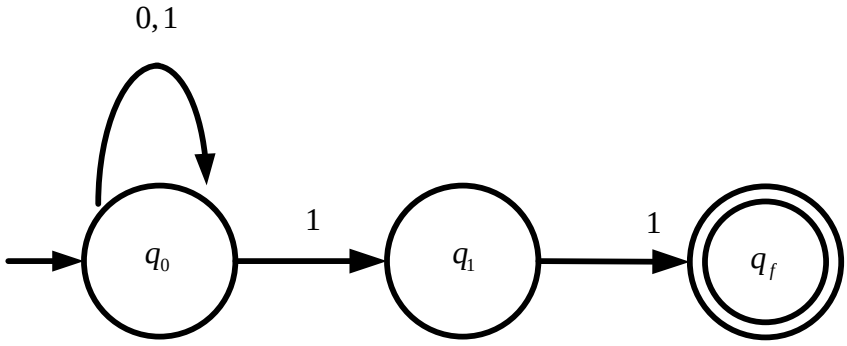

Fig. 1.2. Transition diagram for NFA $(0+1)^*11$

Fig. 1.2 represents the transition diagram of NFA for a regular expression $(0+1)^*11$ and Fig. 1.3 represents the processing of string $w = 1011$.

DEFINITION 1.2.6. Let X be a non-empty set. A fuzzy set A [10] is defined as ordered pair $\{(w, \mu_A(w))\}$ for $\exists w \in X^*$. $\mu_A(w)$ is interpreted as a grade of membership, and it is defined by a membership function $\mu_A : X \to [0,1]$.

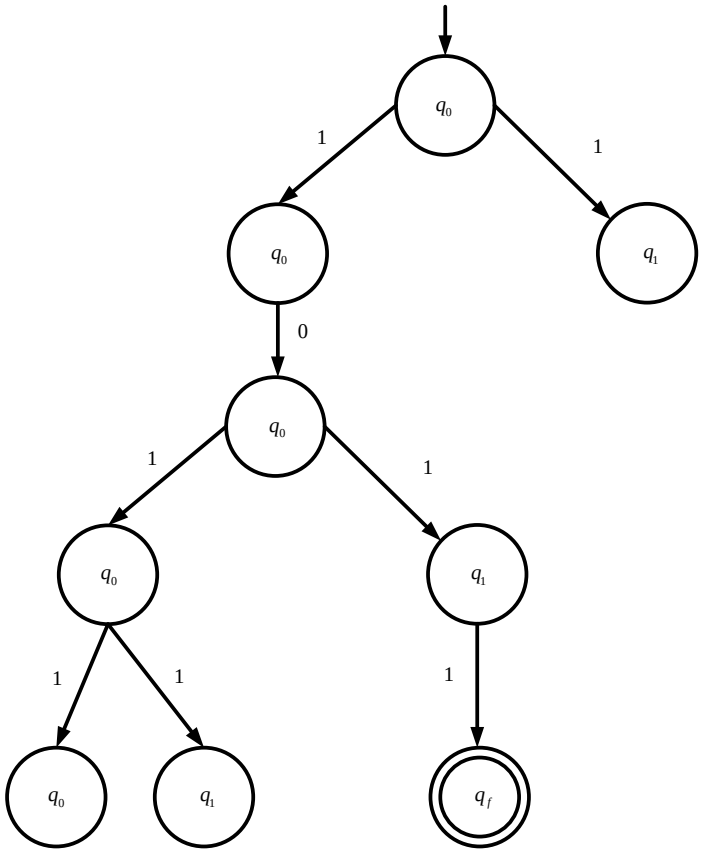

Fig. 1.3. Processing of string $w = 1011$ by NFA

DEFINITION 1.2.7. A lattice-ordered monoid or ℓ – monoid [17] $(L, \wedge, \vee, \otimes, 0, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice with a least element 0 and the greatest element 1,
- $(L, \otimes, e)$ is a monoid with the unit element e,
- For $\forall x \in L$, $x \otimes 0 = 0 \otimes x = 0$,
- For $\forall x, y, z \in L$,

  $x \otimes (y \vee z) = x \otimes y \vee x \otimes z$

  $(x \vee y) \otimes z = x \otimes z \vee y \otimes z$

DEFINITION 1.2.8. Integral ℓ-monoid [17] is a ℓ-monoid in which the unit element e and 1 coincide.

DEFINITION 1.2.9. A fuzzy regular expression [21-22] f_{re} can be recursively defined as follows:

- $\varepsilon,\ \phi, a \in \Sigma$ are fuzzy regular expressions,
- $\lambda r_1 \in f_{re}, \lambda \in L$ and $r_1 \in f_{re}$, and
- If r_1 and r_2 are fuzzy regular expressions, then $r^*, r_1 + r_2, r_1 r_2$ are also fuzzy regular expressions.

All fuzzy regular expressions can be obtained by applying above rules finite number of times.

DEFINITION 1.2.10. The fuzzy language $\| r \|$ [21] represented by the fuzzy regular expression $r \in LR$ can be defined as follows:

- $\| \phi \| = 0$,
- For $x \in X \bigcup \{\varepsilon\};\ \| x \| = \{x/1\}$,
- For $\lambda \in L, r \in LR;\ \| \lambda r \| = \lambda \otimes r$,
- $r_1,\ r_2 \in LR;\ \| r_1 + r_2 \| = \| r_1 \| + \| r_2 \|$,
- $r_1, r_2 \in LR; \| r_1 r_2 \| = \| r_1 \| \| r_2 \|$, and
- $r \in LR; \| r^* \| = \| r \|^*$

DEFINITION 1.2.11. A fuzzy automaton [58] is a quintuple $(Q,\ \Sigma, \sigma, \tau, \delta)$, where

- Q is the set of states,
- Σ represents an alphabet,
- σ is a fuzzy subset of Q (starting fuzzy state),
- τ is the fuzzy set of final states, and
- δ is a fuzzy subset of $Q \times (\Sigma \bigcup \varepsilon) \times Q$.

DEFINITION 1.2.12. The fuzzy language $L(M)$ [18] accepted by the fuzzy automaton M is defined by

For any word $x \in X^*, x = x_1 x_2 ... x_n, \exists x_i \in X$

$L(M)(x) = \sigma \circ \delta_{x_1} \circ \delta_{x_2} \circ ... \circ \delta_{x_n} \circ \tau$

Using composition operator on fuzzy relations [18]

$L(M)(x) = \vee \sigma(q_0) \otimes \delta_{x_1}(q_0, q_1) \otimes \delta_{x_2}(q_1, q_2) \otimes ... \otimes \delta_{x_n}(q_{n-1}, q_n) \otimes \tau(q_n)$

For empty string

$$L(M)(x) = \sigma \circ \tau = \vee(\sigma(q_0) \otimes \tau(q_0))$$

Consider fuzzy finite automata of Fig. 1.4 and $w = 000$

$$\mu(w) = \max(\min(0.7, q_0 \overset{0.2}{\to} q_0 \overset{0.2}{\to} q_0 \overset{0.2}{\to} q_0, 0), \min(0.7, q_0 \overset{0.2}{\to} q_0 \overset{0.2}{\to} q_0 \overset{0.3}{\to} q_1, 1), \min(0.7, q_0 \overset{0.2}{\to} q_0 \overset{0.5}{\to}$$

$$q_1 \overset{0.3}{\to} q_1, 1), \min(0.7, q_0 \overset{0.2}{\to} q_0 \overset{0.5}{\to} q_1 \overset{0.7}{\to} q_0, 0), \min(0.7, q_0 \overset{0.5}{\to} q_1 \overset{0.3}{\to} q_1 \overset{0.3}{\to} q_1, 1), \min(0.7, q_0 \overset{0.5}{\to} q_1 \overset{0.3}{\to} q_1 \overset{0.7}{\to} q_0, 0)$$

$$\min(0.7, q_0 \overset{0.5}{\to} q_1 \overset{0.7}{\to} q_0 \overset{0.5}{\to} q_1, 1)) = 0.5$$

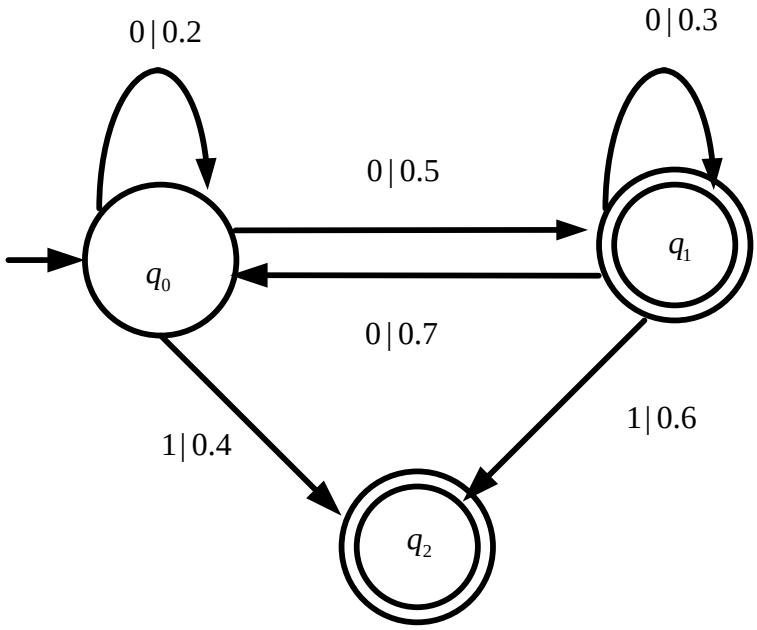

Fig. 1.4. Fuzzy Automata with initial state $\sigma = \{q_0 \mid 0.7\}$ and $\tau = \{q_1 \mid 1, q_2 \mid 0.8\}$

DEFINITION 1.2.13. A fuzzy grammar [59] is a quadruple $G(N, T, P, S)$, where the production rules are of the form $\alpha \overset{\mu}{\to} \beta$ where $\alpha, \beta \in (N \bigcup T)^*, \mu \in [0,1]$.

Example 1.2.4. Fuzzy grammar $G(\{S, A, B\}, \{0,1\}, P, \{S\})$, where production rules are of the form

$$S \overset{0.5}{\to} 1S, \quad S \overset{0.3}{\to} 1A, \quad S \overset{0.4}{\to} 0A, \quad A \overset{0.4}{\to} 0,$$

$$S \overset{0.5}{\to} 1, \quad S \overset{0.6}{\to} 1B, \quad S \overset{0.1}{\to} 0, \quad B \overset{0.5}{\to} 0$$

$$\mu_G(10) = \max(\min(S \overset{0.6}{\to} 1B \overset{0.5}{\to} 10), \min(S \overset{0.3}{\to} 1A \overset{0.4}{\to} 10), \min(S \overset{0.5}{\to} 1S \overset{0.1}{\to} 10)) = \max(0.5, 0.3, 0.1) = 0.5$$

DEFINITION 1.2.14. Shuffle operator [40, 60] is represented by &. It can be inductively defined as follows:

- $x \& \varepsilon = \varepsilon \& x = \{x\}$,
- $ax \& by = \{ax_1 \mid x_1 \in x \& by\} \bigcup \{by_1 \mid y_1 \in ax \& y\}$.

Example 1.2.5: Given $w_1 = 01$ and $w_2 = ab$ then

$w_1 \& w_2 = \{01ab, 0a1b, 0ab1, ab01, a0b1, a01b\}$

DEFINITION 1.2.15. A parallel regular expression [7] can be recursively defined as follows:

- $\varepsilon, \phi, a \in \Sigma$ are regular expressions, and
- If r_1 and r_2 are regular expressions, then $r_1^*, r_1 r_2, r_1 + r_2, r_1 \& r_2$ are also parallel regular expressions.

All parallel regular expressions can be obtained by applying above rules finite many times.

DEFINITION 1.2.16. A parallel finite automaton [7] is septuple $(Q, N, \Sigma, \sigma, \tau, \delta, \gamma)$, where

- Q is the finite set of states and $Q \subseteq 2^N$,
- N is a finite set of nodes,
- Σ represents an alphabet,
- σ is the starting state and $\sigma \in Q$,
- $\tau \subseteq Q$ is the set of final states,
- δ represents the state transition function defined by $Q \times (\Sigma \bigcup \lambda) \to 2^Q$, and
- γ represents the node transition function defined by $2^N \times (\Sigma \bigcup \lambda) \to 2^{2^N}$.

A state in a parallel finite automaton represents a set of nodes. The additional symbol λ in δ and γ represents the path divergence and convergence [32].

Fig. 1.5 and 1.6 represent the parallel finite automaton and its equivalent deterministic finite automaton for parallel regular expression $a \& b^*$.

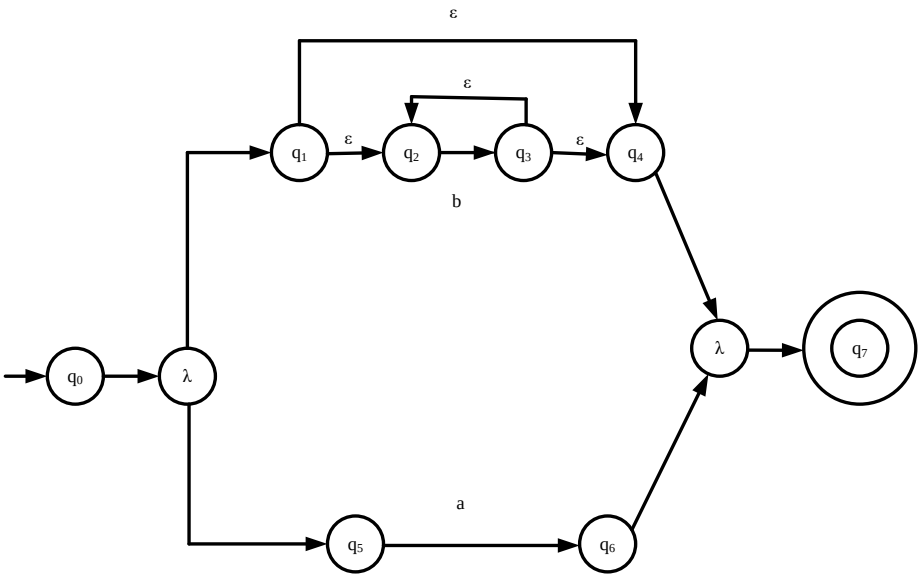

Fig. 1.5. Parallel Finite Automata for parallel regular expression a & b^*.

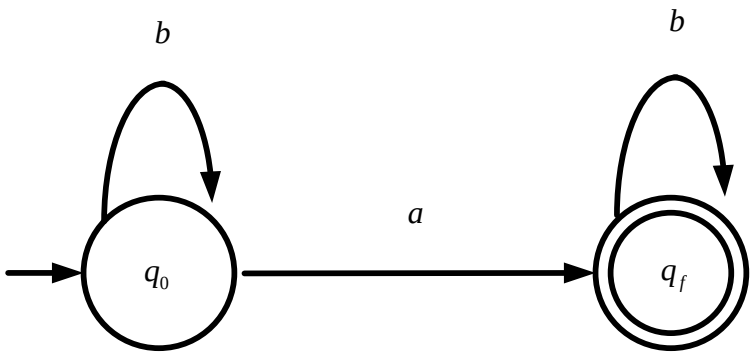

Fig. 1.6. DFA for parallel regular expression a & b^*.

DEFINITION 1.2.17. For regular expression r, assign a subscript to each letter from left to right which will refer its position. The position set $First, Last$ and $Follow$ are computed inductively as follows [61]:

$$First(\phi) = First(\varepsilon) = Last(\phi) = Last(\varepsilon) = \phi$$

$$First(a_i) = Last(a_i) = \{ i \}$$

$$First(r_1 + r_2) = First(r_1) + First(r_2)$$

$$Last(r_1 + r_2) = Last(r_1) + Last(r_2)$$

$$First(r_1^*) = First(r_1)$$

$$Last(r_1^*) = Last(r_1)$$

$$If \quad \varepsilon \in r_1$$

$$First(r_1 r_2) = First(r_1) \bigcup First(r_2)$$

Else

$$First(r_1 r_2) = First(r_1)$$

EndIf

If $\varepsilon \in r_2$

$$Last(r_1 r_2) = Last(r_1) \bigcup Last(r_2)$$

Else

$$Last(r_1 r_2) = Last(r_2)$$

EndIf

$$Follow(\phi) = Follow(\varepsilon) = Follow(a) = \phi$$

$$Follow(r_1 + r_2) = Follow(r_1) \bigcup Follow(r_2)$$

$$Follow(r_1 r_2) = Follow(r_1) \bigcup Follow(r_2) \bigcup Last(r_1) \times First(r_2)$$

$$Follow(r_1^*) = Follow(r_1) \bigcup Last(r_1) \times First(r_1)$$

Example 1.6. Given regular expression $r = a(a+b)^* b$

Regular Expression after assigning position $r = a_1(a_2 + b_3)^* b_4$

$$First(a_1) = Last(a_1) = \{1\}$$

$$First(a_2) = Last(a_2) = \{2\}$$

$$First(a_3) = Last(a_3) = \{3\}$$

$$First(a_4) = Last(a_4) = \{4\}$$

$$First(a_2 + b_3) = Last(a_2 + b_3) = \{2, 3\}$$

$$First((a_2 + b_3)^*) = Last((a_2 + b_3)^*) = \{2, 3\}$$

$$First(a_1(a_2 + b_3)^*) = First(a_1) = \{1\}$$

$$Last(a_1(a_2 + b_3)^*) = Last(a_1) \bigcup Last((a_2 + b_3)^*) = \{1, 2, 3\}$$

$$First(a_1(a_2 + b_3)^* b_4) = First(a_1) = \{1\}$$

$$Last(a_1(a_2 + b_3)^* b_4) = Last(a_1) = \{4\}$$

$$Follow(1) = \{2, 3, 4\}$$

$$Follow(2) = \{2, 3, 4\}$$

$$Follow(3) = \{2, 3, 4\}$$

$Follow(4) = \phi$

1.3 ORGANIZATION OF THE THESIS

The main aim of this thesis is to propose some improvements in the construction of fuzzy automata. The results obtained are discussed in chapter 3 to chapter 5. Complete chapter-wise summary of thesis work is as follows:

- **In Chapter 1**, the motivation of research work, preliminary concepts and, the outline of the thesis has been given.
- **In Chapter 2**, we review the existing literature on fuzzy regular expression, fuzzy grammars, fuzzy automaton and their applications. Mainly, we discuss the survey on conversion of fuzzy regular expressions to fuzzy finite automata, conversion of the finite automaton to a regular expression, and conversion of parallel regular expression to a finite automaton. In the last section of this chapter, we discussed the research gaps in the area of fuzzy grammar and fuzzy automata, and the objectives of this dissertation are explored.
- **In Chapter 3**, a novel algorithm is proposed for carrying out the conversion of parallel fuzzy regular expression to ε-free non-deterministic fuzzy automaton. There exists a variety of shuffle operators, such as weak synchronized shuffling, strong synchronized shuffling, synchronous composition and asynchronous interleaving. In this work, we used asynchronous interleaving shuffle operator. Furthermore, an analysis of the number of states obtained from the fuzzy automaton will be carried out using the proposed algorithm. The proposed algorithm is illustrated by a numerical example. At last, we proved the equivalence between parallel fuzzy regular expression and parallel fuzzy finite automaton.
- **In Chapter 4**, we generalize and extend the concept of transitive closure for the conversion of fuzzy automaton into a fuzzy regular expression. In this work, we propose an algorithm for the conversion of fuzzy automaton into fuzzy regular expression using transitive closure. We support this conversion with the help of a numerical example. Furthermore, we have proven that there exists a fuzzy regular expression *r* corresponding to the fuzzy automaton M using mathematical induction. The main result of this chapter is as follows:

- **Theorem 1.** Given a fuzzy automaton, $M = (Q, X, \delta, \sigma, \tau)$ there exists a fuzzy regular expression r obtained from M using the algorithm *FA*, $FRE(M, r)$ such that $L(M) = L(r)$.

- **In Chapter 5**, Bio-molecular structures (Inverted repeat, Hairpin, Stem and Loop stricture) are represented using linear expressions. We have introduced the concept of fuzzy linear expression. An algorithm is proposed for the conversion of a fuzzy linear grammar to its normal form. In the last section, we have described a new variant of Antimirov's partial derivatives for the conversion of fuzzy regular expression to ε-free fuzzy non-deterministic finite automaton. The membership values in fuzzy automata are assigned using integral lattice-ordered monoids. We have also done the comparison with Stamenkovic and Ćirić approach [17] and found that clearly, the number of states of the ε-free fuzzy automata is less than the number of states resulting from Stamenkovic and Ćirić approach [17].The resulting automaton is an improvement over the existing approaches, and the resulting automaton is ε-free. Further, we prove that non-deterministic fuzzy automaton M is obtained from a fuzzy regular expression r using mathematical induction. The main result of this chapter is as follow:

- **Theorem 2:** Let r be a fuzzy regular expression, then the ε-free fuzzy automata M is obtained using the extended partial derivative approach such that $LR(M) = LR(r)$.

Finally, conclusions and future scope are presented in **Chapter 6**, followed by references.

Chapter-2

LITERATURE SURVEY

In this chapter, we will review the existing literature on regular expressions, finite automata, fuzzy regular expressions, fuzzy automata, and their applications. Mainly, we will discuss the conversion of regular expression to/from a finite automaton, conversion of fuzzy regular expression to/from a fuzzy finite automaton, and conversion of parallel regular expression to a finite automaton. In the last section of this chapter, we will discuss the research gaps in the area of fuzzy grammar, fuzzy automata, and the objectives of this dissertation are explored.

2.1 REVIEW OF LITERATURE

Zadeh [1] proposed the concept of fuzzy set theory in 1965. The notion of fuzzy languages and fuzzy automata was first proposed in 1968 by Santos [2]. Fuzzy automaton is an automaton counterpart of fuzzy grammar. Since 1960, the membership values in fuzzy automata and languages have been considered from the Gödel structure [14]. To enhance the processing ability of fuzzy automata, the membership values are extended to algebraic structures, such as lattice-ordered monoids [17-22, 62], complete residuated lattices [23-24]. A fuzzy automaton is an instance of weighted automata [63] with values taken from a lattice-ordered monoid rather than a more general semiring structure. Asveld [64] proved that fuzzy regular languages are closed under union, intersection, concatenation and Kleene closure. Wee and Fu [8] proposed the mathematical formulation of fuzzy automata.

Mizumoto et al. [65] introduced the concept of formal grammar with weights and classified them into probabilistic, fuzzy and weighted grammars. Doostfatemec and Kremer [66] proposed the concept of augmented transition function for the computation of membership of an element. They computed the membership value of any state by considering the weight of transition and membership value of the previous state. Further, they introduced the concept of multi-membership resolution function named as pre-defuzzification for final as well as non-final states. Furthermore, they also combined the augmented transition function and multi-membership resolution function to extend the fuzzy automata with additional two parameters.

Jin and Li [67] defined lattice-valued finite automata and lattice-valued regular grammar on the basis of depth-first and breadth-first. In lattice-valued regular grammar, if the lattice domain is distributed, then both depth-first and breadth-first are equivalent. Further, they also proved that lattice-valued finite automata and lattice-valued regular grammar are equivalent. Later, they proved that lattice-valued finite automata, lattice-valued deterministic finite automata, lattice-valued regular grammar and lattice-valued deterministic regular grammars are equivalent only in the depth-first way. Jun and Kavikumar [68] proposed the novel concept of bipolar fuzzy sets which is an extension of fuzzy sets. In bipolar fuzzy sets, the membership degree lies between [-1 to 1] rather than [0 to 1]. Kavikumar et al. [69] introduced the notion of fuzzy entire sequence space.

2.2 SURVEY ON CONVERSION OF REGULAR EXPRESSION TO FINITE AUTOMATON

There exist various approaches for the conversion of regular expression to finite automaton such as Thompson [70], Sippu and Soisalon-Soininen [71], Glushkov [29], Berry and Sethi [72], partial derivative [54], McNaughton and Yamada [30], and follow automata [73]. Table 2.1 describes a summary of various existing approaches for the conversion of regular expression to a finite automaton.

Table 2.1: Summary of various research finding for the conversion of regular expression to a finite automaton.

Author's	**Notable Findings**
Thompson [70]	• Proposed an algorithm for the conversion of regular expression to ε-NFA in linear time. • It works recursively by splitting the regular expression into sub-expressions, and individual NFA's are constructed from sub-expressions with the help of well-defined rules. • Finally, the obtained NFA's of sub-expressions are merged into a single equivalent NFA.
Sippu and Soisalon-Soininen [71]	• They have extended Thompson approach, but the NFA obtained is having lesser number of states than Thompson's construction.

Author's	Notable Findings
Glushkov [29]	• Proposed the concept of positional automata. • The obtained NFA is ε-free and has exactly $m+1$ state, where m is the number of instance of symbols in the regular expression.
McNaughton and Yamada [30]	• He proposed an approach to construct positional automata from regular expressions in quadratic time.
Bruggemann-Klein [74]	• Using his approach positional automata can be obtained in quadratic time.
Berry and Sethi [72]	• Proposed an algorithm to convert the regular expression into equivalent NFA in cubic time.
Sempere [75]	• Introduced the concept of linear expression for representing the linear context-free languages. • Proposed algorithms for the conversion of linear grammar to/from linear expression.
Champarnaud et al. [76]	• Introduced the concept of two-sided derivatives • They designed a polynomial construction of finite recognizer for Hairpin structure.
Champarnaud and Ziadi [53]	• Proposed an improved algorithm for the conversion of a regular expression into equivalent NFA using partial derivative with complexity $O(n^2)$.
Ilie and Yu [73]	• They proposed the concept of follow automata. • They designed an approach using which positional automata can be converted into follow automata using follow equivalence.

2.2.1 NUMERICAL EXAMPLES OF VARIOUS APPROACHES

In this subsection, Thompson construction [70], Sippu and Soisalon-Soininen [71] and Positional Automata [29] are described with numerical examples.

Example 2.1. Construction of NFA for $0(0|1)^*1$ using Thompson's construction.

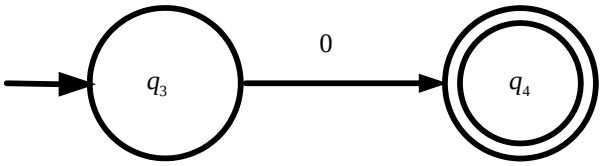

Fig. 2.1. NFA for regular expression $r_2 = 0$ using Thompson's construction

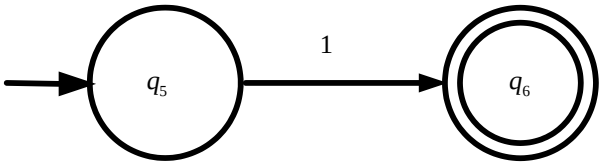

Fig. 2.2. NFA for regular expression $r_3 = 1$ using Thompson's construction

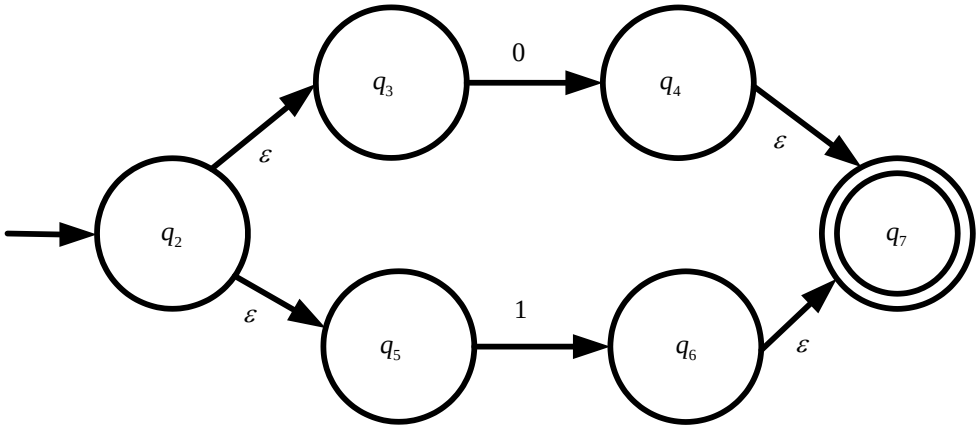

Fig. 2.3.NFA for regular expression $r_4 = 0|1$ using Thompson's construction

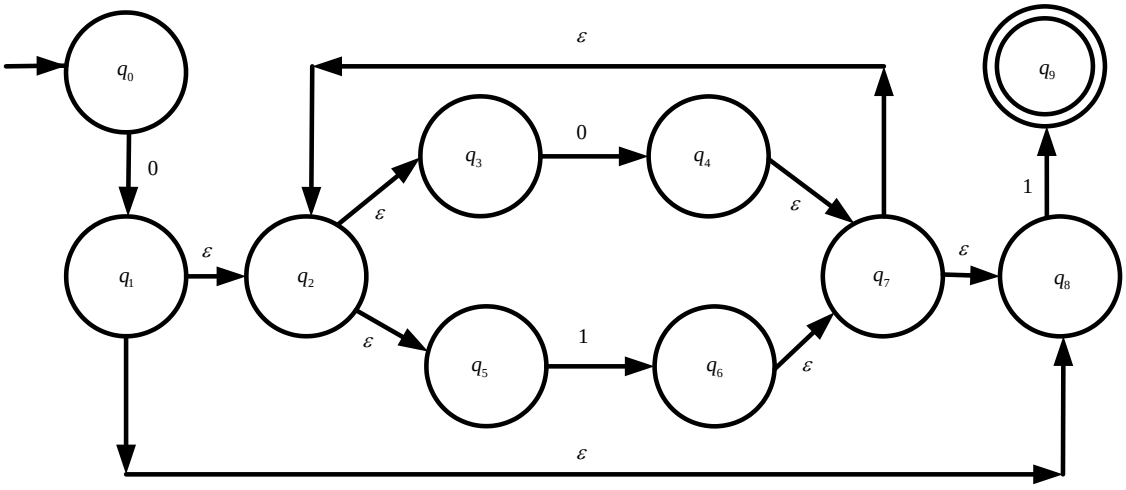

Fig. 2.4. NFA for regular expression $0(0|1)^*1$ using Thompson's construction

Example 2.2. Construction of NFA for $0(0|1)^*1$ using Sippu and Soisalon-Soininen's construction.

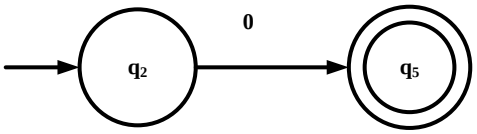

Fig. 2.5. NFA for regular expression $r_2 = 0$ using Sippu and Soisalon-Soininen's construction.

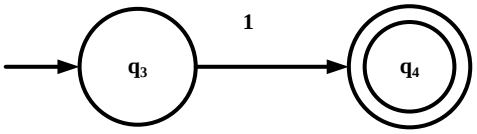

Fig. 2.6. NFA for regular expression $r_3 = 1$ using Sippu and Soisalon-Soininen's construction.

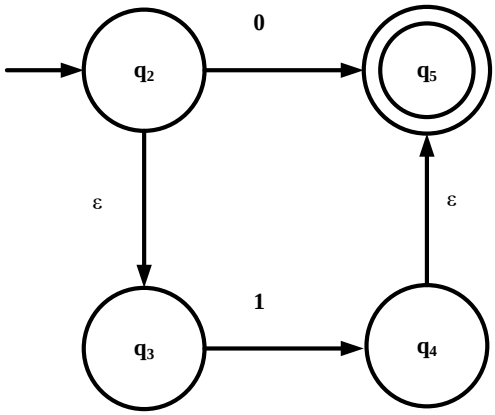

Fig. 2.7. NFA for regular expression $r_4 = 0|1$ using Sippu and Soisalon-Soininen's construction.

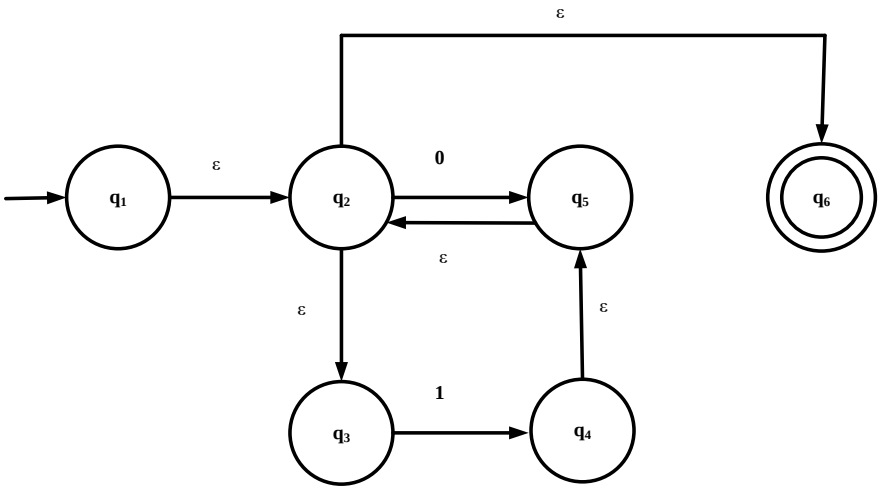

Fig. 2.8. NFA for regular expression $r_5 = (0|1)^*$ using Sippu and Soisalon-Soininen's construction.

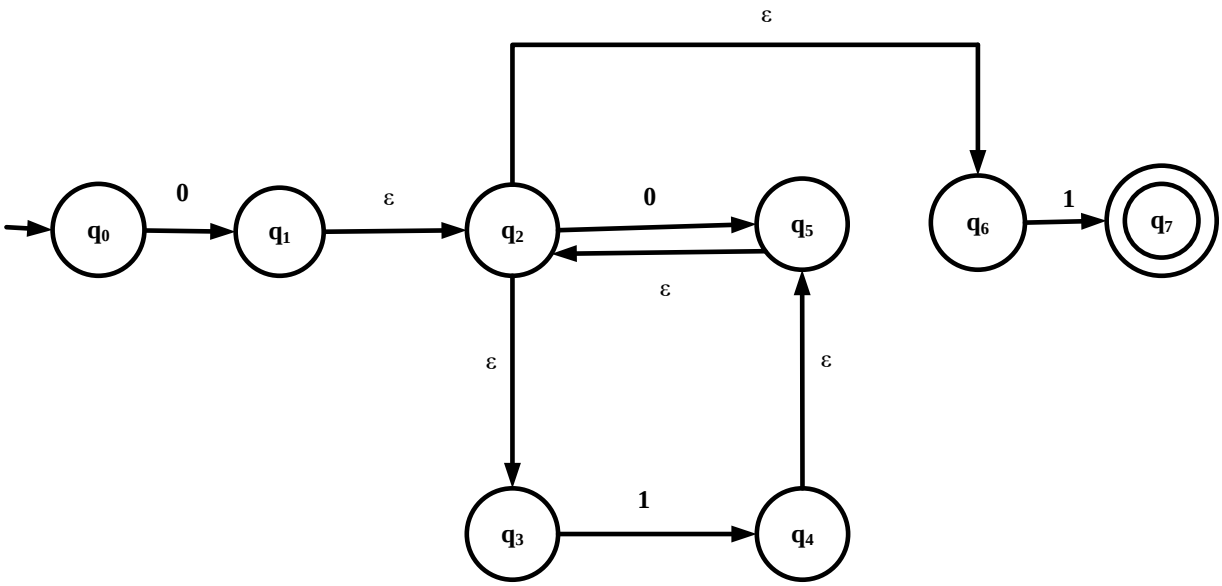

Fig. 2.9. NFA for regular expression $r = 0(0|1)^*1$ using Sippu and Soisalon-Soininen's construction.

Example 2.3. Construction of NFA for $0(0|1)^*1$ using Positional Automata.

Given regular expression $r = 0(0|1)^*1$

Regular expression after assigning position $r = 0_1(0_2|1_3)^*1_4$

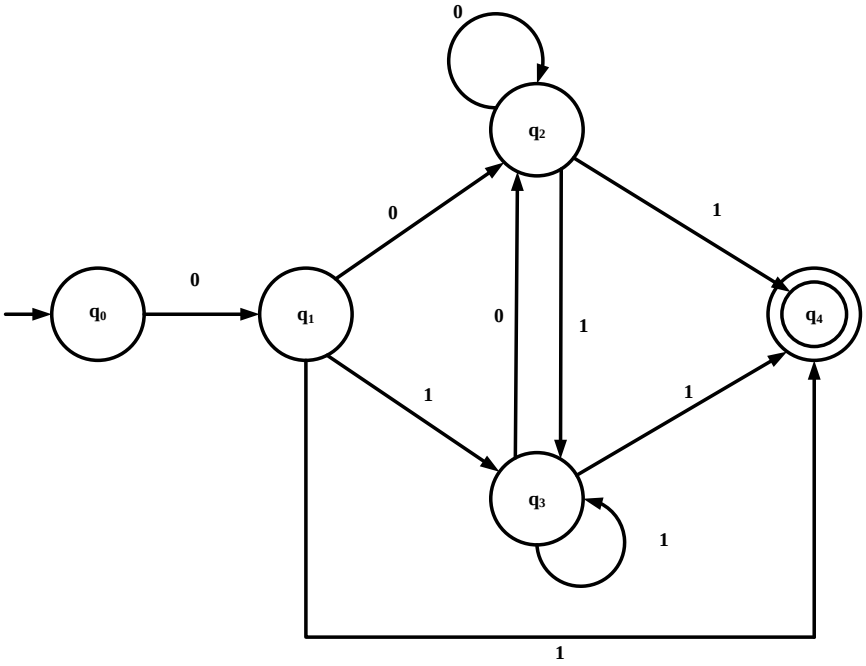

Fig. 2.10. NFA for regular expression $r = 0(0|1)^*1$ using Positional Automata.

Using position automata:

$First = \{1\}$,

$Last = \{4\}$,

$Follow(1) = \{2,3,4\}$,

$Follow(2) = \{2,3,4\}$,

$Follow(3) = \{2,3,4\}$, and

$Follow(4) = \{\phi\}$.

Fig. 2.10 represents the Positional automata for a regular expression $r = 0(0|1)^*1$.

2.3 SURVEY ON CONVERSION OF FINITE AUTOMATA TO REGULAR EXPRESSIONS

There exist various approaches for the conversion of finite automata to regular expressions such as Transitive Closure [41], Brzozowski [42] and state elimination method [43]. Table 2.2 describes various existing approaches for the conversion of finite automaton to a regular expression.

Table 2.2: Summary of various approaches used for the conversion of the finite automaton to a regular expression.

Author's	**Notable Findings**
Kleene [41]	• Proposed a novel technique known as a transitive closure method for the conversion of deterministic finite automata to regular expressions.
Brzozowski [42]	• Defined the derivative of regular expressions as an extension to Kleene method [41].
Du and Ko [43]	• Proposed the state removal method for the conversion of deterministic finite automata to regular expressions.
Neumann [44]	• Examined three approaches for the transformation of finite automata to regular expressions. • Followings are the keypoints of their work: i. Transitive closure: Simple and easy to implement. ii. State removal: Difficult to implement but easy to recognize the regular expression manually. iii. Brzozowski algebraic: Easy to implement but best-suited for recursive languages.

Author's	Notable Findings
Gruber and Holzer [61]	• Proposed an approach for the transformation of finite automaton to a regular expression using state elimination method.

2.4 SURVEY ON CONVERSION OF FUZZY REGULAR EXPRESSION TO FUZZY FINITE AUTOMATON

Stamenkovic and Ćirić [17] proposed an elegant method for the conversion of fuzzy regular expressions into fuzzy automata based on the concept of position automaton [29-30]. They treated the scalar appearing in the fuzzy regular expression as the extended alphabet. The major issue in the proposed construction model is that the number of states remains same as those of conventional approaches. Further Ćirić et al. [23] extended their previous work of [17] to reduce the number of states of fuzzy finite automata using fuzzy equivalence. First, they defined left and right invariant fuzzy crisp equivalences. Furthermore, they proposed a reduced variant of fuzzy finite automata (FFA) which is having less number of states than FFA by using right invariant crisp equivalences. At last, they proved the results of a reduced variant of FFA by taking membership grades in an integral lattice ordered monoid. Stamenkovic et al. [77] extended their previous work of [17] to reduce the number of states of fuzzy finite automata using fuzzy quasi-orders. Initially, they defined left and the right invariant of fuzzy quasi-orders, then they proved that both fuzzy quasi-order and fuzzy equivalences are equivalent in terms of state reduction, but the left invariant fuzzy quasi-orders and right invariant fuzzy quasi-orders shows better reduction results than the left and right invariant fuzzy equivalences. Mendivil and Garitagotia [18] extended the Stamenkovic and Ćirić [17] approach and removed the ε-transition from the fuzzy automata.

Kuske [31] proposed a similar approach for the weighted regular expression over a semiring. Kumar and Kumar [19, 78] outlined a method for the conversion of fuzzy regular expressions into fuzzy automata based on the follow automata [73]. Li and Pedrycz [21] introduced the concept of integral lattice-ordered monoid in fuzzy regular expressions. Li et al. [22] introduced the concept of ε- move in fuzzy automata. Mohri

[79], Allauzen and Mohri [80] proposed an approach for the ε-removal in weighted automata.

Astrain et al. [81] defined fuzzy automata by considering the empty string and proved that it generates the same set of languages of fuzzy automata. Further, they defined the fuzzy measure between strings using string alignment and fuzzy edit operations. Ravi and Choubey [82] proposed the concept of interval-valued fuzzy regular expressions and its corresponding interval-valued fuzzy deterministic and non-deterministic finite automaton. Furthermore, they found that the interval-valued fuzzy final states model is more efficient than interval-valued fuzzy transitions model.

Thompson [70] proposed a method for constructing deterministic fuzzy finite automata and its corresponding unambiguous grammar for the fuzzy regular expression. Furthermore, they also defined the normal form for the fuzzy regular grammar production using the max-min method. Cao and Ezawa [48] proposed the concept of non-deterministic fuzzy automata and recognized the corresponding fuzzy language. They proposed non-deterministic fuzzy automata with or without null moves. Furthermore, they proved that deterministic and non-deterministic fuzzy automata with or without null moves recognize the same class of fuzzy language.

Chaudhari and Komejwar [83] proved that fuzzy regular grammar and fuzzy finite automata are equivalent. Lee [84] proposed a method to convert fuzzy automata to canonical fuzzy automata. He further defines an algorithm to minimize the states of fuzzy automata in the canonical form. Peeva and Zahariev [85] defined finite state fuzzy machines. They proposed an algorithm and developed its corresponding software for computing the behavior (i.e. behavior matrix) and establishing the states equivalence. Further, they observed the behavior matrix and obtained the reduced and minimal finite state fuzzy machine.

Topencharov and Peeva [86] proposed novel algebraic algorithms for reduction, minimization and equivalence of fuzzy finite automata. Furthermore, in [87], Peeva extended his work of [86] and proposed a novel approach for reduction, minimization and equivalence of finite automata over semirings. In this work, the author applied the approach on deterministic finite automata, non-deterministic finite automata, stochastic finite automata and fuzzy automata. Cheng and Mo [88] introduced fuzzy Mealy finite state automata and devised the minimal algorithm for fuzzy Mealy finite state automata.

Table 2.3 describes the various existing approaches for the conversion of fuzzy regular expression to fuzzy finite automaton.

Table 2.3: Summary of significant research finding for the conversion of fuzzy regular expression to fuzzy finite automaton.

Author's	Notable Findings
Stamenkovic and Ćirić [17]	• Proposed a method for the conversion of fuzzy regular expression into fuzzy automaton based on the position automata.
Mendivil and Garitagoitia [18]	• Proposed approach removes the ε-transition from the fuzzy automata. • This approach is an extension of the work done by Stamenkovic and Ćirić [17] for the conversion of fuzzy regular expression to fuzzy automaton.
Li et al. [21]	• Introduced the concept of ε - move in fuzzy automata.
Kuske [31]	• Proposed a method for the conversion of weighted regular expression into weighted automata over a semiring.
Yamamoto [45-46]	• Proposed an approach for the conversion of semi-extended regular expressions to non-deterministic finite automata (NFA).
Thomason and Marinos [47]	• Proposed a method for constructing deterministic fuzzy finite automata and its corresponding unambiguous grammar for the fuzzy regular expression.
Cao and Ezawa [48]	• Proposed the concept of non-deterministic fuzzy automata and recognize the corresponding fuzzy language. • Proposed the two variant of non-deterministic fuzzy automata with or without null moves.
Kumar and Kumar [20,78]	• Proposed a method for the conversion of fuzzy regular expression into fuzzy automaton based on the follow automata.
Astrain et al. [81]	• Defined fuzzy automata considering empty string. • Proved the fuzzy automata equivalence with or without empty string.

Author's	Notable Findings
Ravi and Choubey [82]	• Proposed interval-valued fuzzy regular expressions and its corresponding interval-valued fuzzy deterministic and non-deterministic finite automata. • Proved the equivalence between interval-valued fuzzy regular language and interval-valued fuzzy deterministic finite automata and fuzzy non- deterministic finite automata. • Interval-valued fuzzy final states model is more efficient than interval -valued fuzzy transitions model.
Chaudhari and Komejwar [83]	• Proved the equivalence between fuzzy regular grammar and fuzzy finite automata.

2.5 SURVEY ON CONVERSION OF FUZZY FINITE AUTOMATA TO FUZZY REGULAR EXPRESSIONS

Omlin et al. [89] designed an approach for the representation of fuzzy finite automata in recurrent neural networks. The recurrent neural network is built from the architecture of DFA encoding. This recurrent neural network construction of fuzzy finite automata helps in recognizing the fuzzy regular language.

Blanco et al. [90] designed an approach for the representation of fuzzy finite automata in two-layer neural networks. From the set of fuzzy strings, the neural network is trained which further extracts fuzzy automata. This two-layer neural network construction of fuzzy finite automata helps in recognizing fuzzy regular grammar. Table 2.4 describes the various existing approaches for the conversion of fuzzy finite automaton to fuzzy regular expressions.

Table 2.4: Summary of various research finding for the conversion of fuzzy finite automaton to fuzzy regular expression.

Author's	Notable Findings
Omlin et al. [89]	• Proposed an approach for representation of fuzzy finite automata in recurrent neural networks.

	• This recurrent neural network construction of fuzzy finite automata helps in recognizing the fuzzy regular language.
Le [91]	• Designed the non-deterministic finite automata for mono sound waves of the keywords of speech. • Further, the author designed a pattern matching procedure machine which helps in identification of keywords from the automata.
Blanco et al. [90]	• Designed an approach for the representation of fuzzy automata in two-layer neural network. • Training of this two-layer neural network is carried out by a set of fuzzy strings. • This two-layer neural network construction of fuzzy finite automata helps in recognizing fuzzy regular grammar.

2.6 SURVEY ON TRANSFORMATION OF PARALLEL REGULAR EXPRESSION TO FINITE AUTOMATON

Parallel regular expressions were introduced by Garg [92] in the late 1990s. Subsequently, the transformation of parallel regular expressions to finite automata was established by various researchers in the references [7, 32-33, 37-39, 93]. Table 2.5 describes various existing approaches for the conversion of parallel regular expressions to finite automata.

Table 2.5: Summary of various research finding for the conversion of parallel regular expression to finite automaton .

Author's	**Notable Findings**
Estrade et al., Estrade [32-33]	• Proposed an algorithm for the conversion of parallel regular expression to a parallel finite automaton by including shuffle operator. • Discussed the serialized form of parallel finite automata. • Proved the equivalence between parallel finite automata and finite automata.

Author's	Notable Findings
Kumar and Verma [37]	• Proposed a novel approach for obtaining non-deterministic finite automata (null-free) from parallel regular expression by using shuffle operator.
Kumar and Verma [38]	• Author's extended their work of [39] and proposed a novel approach for obtaining non-deterministic finite automaton (null-free) from parallel regular expression.
Stotts and Pugh [8]	• Introduced a composition rule for obtaining parallel finite automata using interleaving operator. • The class of language generated by parallel finite automata using interleaving operator is equivalent to Petri-net languages.
Gelade et al. [93]	• Author's discussed the decidability problems for basic schema languages of XML such as DTD, XSD and Relax NG with an extended regular expression which consider numerical occurrence constraint and interleaving.
Gelade [39]	• The author proposed an approach for the conversion of extended regular expression using interleaving, intersection and counting operator to a regular expression, deterministic finite automata and non-deterministic finite automata. • The author discussed the tight lower bound exponential complexity for all these conversions.

2.7 APPLICATIONS OF FUZZY LANGUAGES AND FUZZY AUTOMATA

Ali and Amalarethinam [94] showed an application of deterministic fuzzy multi-tape finite automata for recognizing the day to day activities of human being. Tiwari and Sharan [95] proposed the concept of direct, cascade and wreath products of rough finite state machines. Further, they have discussed the algebraic properties and relationship

among these properties. Sharan and Tiwari [96] further extended their work of [95] by proposing the concept of different products of mealy type rough finite state machines and discussed their properties.

Qui [58] introduced the concept of fuzzy finite automata with the bi-fuzzy property. The author describes the notions of a bi-fuzzy family of sub automata, bi-fuzzy source operator, bi-fuzzy successor operator and presented topological properties of bi-fuzzy fuzzy finite automata.

Kartikeyan and Rajasekar [97] proposed γ-Synchronised fuzzy automata. Furthermore, in [98] they have applied their concept of [97] of γ-Synchronized fuzzy automata in finding the shortest optimal path for petrol passing pipelines connecting four cities. The main aim of their work was that if we start from one city, the petrol should reach every city at least once and the city where it ends should be with minimal flow capacity and maintenance charge.

Yu [99] introduced the concept of fuzzy L-languages and explored its properties. He measured the fuzzy entropy with respect to L-system and discussed the relationship between fuzzy L-languages, fuzzy grammar languages and ordinary L-systems. Reyneri [100] used the concept of fuzzy finite automata in control applications. The author proposed the fuzzy finite automata for neuro-fuzzy systems which is an extension of finite automata for neuro-fuzzy systems. Reyneri and Morano [101] extended their previous work of [100] for applying the concept of fuzzy finite automata in neuro-fuzzy systems. In this work, they applied the concept of neuro-fuzzy system finite automata for observing the gait control of the legged robot. Kovacs [102] used the concept of fuzzy reasoning and fuzzy automata in behavior based control structures. Giles et al. [103] proposed an algorithm for constructing augmented recurrent neural network which cryptograph the fuzzy finite automata and further recognizes its fuzzy regular language. Mamdani [104] integrated the concept of fuzzy logics in the control of the dynamic plant. In this work, the author developed a fuzzy framework in the laboratory based steam engine. Asai and Kitajima [105] used the concept of fuzzy logic in optimizing control of multi-model systems. They used fuzzy mealy and moore machine as an optimizing controller in optimizing the control of multi-model systems.

De et al. [106] discussed the application of intuitionistic fuzzy sets for medical diagnosis. Steimann and Adlassnig [107] proposed a framework for intelligent state monitor using

fuzzy logic for diagnosing the medical condition of the patient. In a medical fuzzy system, the patient state or condition, the input to the patient (medicine) and the state transition functions (i.e. transition from one condition to another) are all based on the fuzzy concepts. Further, in [108], they extended their work of medical diagnosis using fuzzy logic. In this work, they worked on scoring, monitoring and consultation of diagnosis using fuzzy logic.

Wee and Fu [8] developed a learning model using fuzzy automata. The proposed fuzzy learning model helps to evaluate the performance of teacher and student. Kakoty et al. [26] developed an application to determine the expertise level of learner in an e-learning environment using fuzzy logic.

Thomason [109] applied the concept of fuzzy finite automata in pattern recognition for recognizing the pattern consisting of imprecision or vagueness. Gao and Ovaska [110] used the concept of fuzzy logic for motor fault diagnosis. Le [91] designed the non-deterministic finite automata for mono sound waves for the keywords in the speech. Further, a pattern matching procedure machine was designed for identification of keywords from the automata. Wu et al. [111] integrated the concept of fuzzy logic in image processing. They proposed the concept of fuzzy automata and fuzzy operators for recognization of the images.

Mateescu et al. [112] compared the mealy and moore fuzzy automation models for lexical analysis and found that moore machine is more efficient than mealy machines in the implementation of lexical analyzers. They proposed a novel lexical analyzer tool named as FLEX, which is a fuzzy extension of classical LEX tool [113]. Becchi and Crowley [114] proposed an efficient extended finite state automata for Intrusion detection system using Perl-compatible regular expression. Soni et al. [115] used the action grammar for prototype design system.

Asveld [116] proposed the concept of fuzzy context-free K-grammars for recognizing incorrect strings by finding a finite way to generate the incorrect strings. He proved that limited grammatical errors and fuzzy context-free K- grammars are equivalent to context-free grammars in term of closure properties. Furthermore, the author recognized fuzzy context-free K-grammars using CYK algorithm [117]. Schinder et al. [118] and Inui et al. [119] proposed an approach for recognizing erroneous strings generated by the context-free grammar and context-sensitive grammar, respectively using fuzzy sets.

Bucurescu and Pascu [120] introduced the variants of pushdown automata namely fuzzy pushdown automata and B-fuzzy pushdown automata. Fuzzy pushdown automata and B-fuzzy pushdown automata are the automaton counterpart of context-free language and context-sensitive language, respectively. Weights of fuzzy pushdown automata are assigned in the range between 0 to 1, whereas weights of B-fuzzy pushdown automata are assigned using Boolean algebra. Moraga [121] proved the equivalence between fuzzy context-free languages and fuzzy pushdown automata, by measuring the membership grade depending on weights of a monoid. Hopcroft et al. [40] proved that pushdown automata can be constructed from context-free grammar and vice-versa. Xing [122] introduced the novel concept of one-stack fuzzy push down automata and multi-stack fuzzy pushdown automata. He proved that languages generated by above automaton lies in the family of fuzzy context-free grammars. Arora et al. [123] constructed an algorithm to convert state chart diagram to finite state automata.

2.8 GAPS IN LITERATURE

Based on the literature survey, some of the research gaps identified in the proposed research work of improvements in the construction of fuzzy automata are as follows:

I. In literature study, Stamenkovic and Ćirić [17] proposed an approach for the construction of fuzzy automata from fuzzy regular expressions using the concept of positional automata. State complexity is the major issues in the classical automata theory and fuzzy automata. Therefore, there is a need for an efficient method for the construction of smaller fuzzy automaton from fuzzy regular expression.

II. It has also been observed that there is almost no prior work on additional operators like intersection and shuffle operators in the fuzzy automata. Hence, there is a need of working on fuzzy regular expressions with these additional operators.

2.9 OBJECTIVES

Following are the objectives of this dissertation:

1. To analyze various methodologies existing in the field of fuzzy finite automata.
2. To propose a technique for the transformation of fuzzy regular expression to fuzzy automata.

3. To propose a technique for the transformation of fuzzy regular expression with an additional operator to fuzzy automata.

In addition to the above-mentioned objectives, following objectives are also achieved:

1. Representation of Bio-molecular structure (Inverted repeats, Hairpin, Stem and Loop) using linear expressions.
2. Design of an approach for the conversion of fuzzy automata to fuzzy regular expressions using transitive closure.
3. Introduced the concept of fuzzy linear expressions to represent a fuzzy context-free grammar.

Chapter-3

CONVERSION OF PARALLEL FUZZY REGULAR EXPRESSION TO EPSILON-FREE FUZZY AUTOMATON

3.1 INTRODUCTION

The concept of fuzzy languages was introduced by Santos [2]. Fuzzy automata are a generalization of the classical automata, closing the gap between classical automata theory and natural languages [14].The inclusion of shuffle operators in the regular expression is known as a parallel regular expression. It appears in various fields of computer science such as concurrency of processes, process algebra, multipoint communication, plan recognition, *XML* schema language *Relax NG* and modeling and verification of systems [32-36, 124].

Motivated by the applications of fuzzy automata and parallel regular expressions, in this chapter, we propose a new mathematical model that involves fuzziness in parallel finite automata. Stamenkovic and Ćirić [17] developed an approach using the concept of position automata [29-30] for obtaining fuzzy automata from fuzzy regular expressions. The concept of position automata was proposed independently by Glushkov [29], McNaughton and Yamada [30]. Position automata are used to convert a regular expression into a ε-free non-deterministic finite automaton. Positions are assigned to each letter of the regular expression; then using the concept of first, last and follow, a regular expression is converted into a ε-free non-deterministic finite automaton. In this chapter, a novel algorithm is proposed for carrying out the conversion of parallel fuzzy regular expressions to ε-free non-deterministic fuzzy automata. There exists a variety of shuffle operators, such as weak synchronized shuffling, strong synchronized shuffling, synchronous composition and asynchronous interleaving.

In this algorithm, we have used asynchronous interleaving shuffle operator. Furthermore, an analysis on the number of states obtained in the fuzzy automaton will be carried out using the proposed algorithm.

Results of this chapter have been published in the Computer Journal, 59(9), 1383-1391, 2016.

A numerical example also illustrates the proposed algorithm. At last, we proved the equivalence between parallel fuzzy regular expressions and parallel fuzzy finite automata.

3.2 PRELIMINARIES

DEFINITION 3.1 A parallel fuzzy regular expression f_{pre} can be recursively defined as follows:

- $\varepsilon, \phi, a \in \Sigma$ are fuzzy parallel regular expressions.
- $\lambda r_1 \in f_{pre}, \lambda \in L$ and $r_1 \in f_{pre}$
- If r_1 and r_2 are fuzzy parallel regular expressions, then $r^*, r_1 r_2, r_1 + r_2, r_1 \& r_2$ are also fuzzy parallel regular expressions.

All fuzzy parallel regular expressions can be obtained by applying above rules finite many times.

DEFINITION 3.2 A parallel fuzzy finite automaton is septuple $(Q, N, \Sigma, \sigma, \tau, \delta, \gamma)$, where

- Q is the finite set of states and $Q \subseteq 2^N$,
- N is a finite set of nodes,
- Σ represents an alphabet,
- σ is a fuzzy subset of Q (starting fuzzy state),
- τ is the fuzzy set of final states,
- δ represents the state transition function, and it is a fuzzy subset of $Q \times (\Sigma \bigcup \lambda) \to 2^Q$,
- γ represents the node transition function, and it is a fuzzy subset of $2^N \times (\Sigma \bigcup \lambda) \to 2^{2^N}$

3.3 PARALLEL FINITE AUTOMATA

Estrade et al. [32] investigated the equivalence between parallel finite automata and shuffle operation on regular languages. They designed a software tool, FLAT [125], for the conversion of parallel regular expressions to finite automata. In parallel finite automata, a state is represented by a set of active states. They have used λ to denote the divergence and convergence from a node. Parallel fuzzy finite automata are similar to

parallel finite automata, except that the membership value is associated with each transition. This membership value is chosen from a lattice-ordered monoid.

Example 3.1. Given parallel fuzzy regular expression $r = (0.4yx \ \& \ 0.8y)0.2x$

Let $\mu = 0.4,\ \nu = 0.8,\ \kappa = 0.2,\ Y = \{\mu, \nu, \kappa\}$, Extended alphabet = $\{\mu, \nu, \kappa, x, y\}$, $r_1 = (\mu yx \& \nu y)\kappa x,$

$\bar{r_1} = (\mu_1 y_2 x_3 \& \nu_4 y_5)k_6 x_7$

Using position automata:

Follow (1) = {2},

Follow (2) = {3},

Follow (4) = {5}, and

Follow (6) = {7}.

λ represent divergence and convergence at q_0 to (q_1, q_5) and (q_4, q_7) to q_8.

For each transition $(q_{i,} a, q_j) \ \Lambda \ a \in Y$, convert it into $(q_{i,} \in\mid a, q_j)$

Table 3.1: States transition function for parallel fuzzy finite automaton $(Q,\ N, \Sigma, q_0/1, q_f/1, \delta, \gamma)$

Source State	**Input**	**Output**	**Target State**
$[q_0]$	λ	-	$[q_1, q_5]$
$[q_1, q_5]$	ε	0.4	$[q_2, q_5]$
$[q_1, q_5]$	ε	0.8	$[q_1, q_6]$
$[q_2, q_5]$	y	1	$[q_3, q_5]$
$[q_2, q_5]$	ε	0.8	$[q_2, q_6]$
$[q_1, q_6]$	ε	0.4	$[q_2, q_6]$
$[q_1, q_6]$	y	1	$[q_1, q_7]$
$[q_1, q_7]$	ε	0.4	$[q_2, q_7]$
$[q_2, q_6]$	y	1	$[q_3, q_6]$
$[q_2, q_6]$	y	1	$[q_2, q_7]$
$[q_3, q_5]$	x	1	$[q_4, q_5]$
$[q_3, q_5]$	ε	0.8	$[q_3, q_6]$
$[q_3, q_6]$	x	1	$[q_4, q_6]$
$[q_3, q_6]$	y	1	$[q_3, q_7]$

Table 3.1 (Continue)…			
Source State	**Input**	**Output**	**Target State**
$[q_2, q_7]$	y	1	$[q_3, q_7]$
$[q_4, q_5]$	ε	0.8	$[q_4, q_6]$
$[q_4, q_6]$	y	1	$[q_4, q_7]$
$[q_3, q_7]$	x	1	$[q_4, q_7]$
$[q_4, q_7]$	λ	-	$[q_8]$
$[q_8]$	ε	0.2	$[q_9]$
$[q_9]$	x	1	$[q_f]$

For each transition $(q_i, a, q_j) \wedge a \in (\sum \cup \varepsilon)$, convert it into $(q_i, a\,|1, q_j)$.

Each state is represented by a set of active nodes. Table 3.1 depicts different states of the parallel fuzzy finite automata as shown in Fig 3.1.

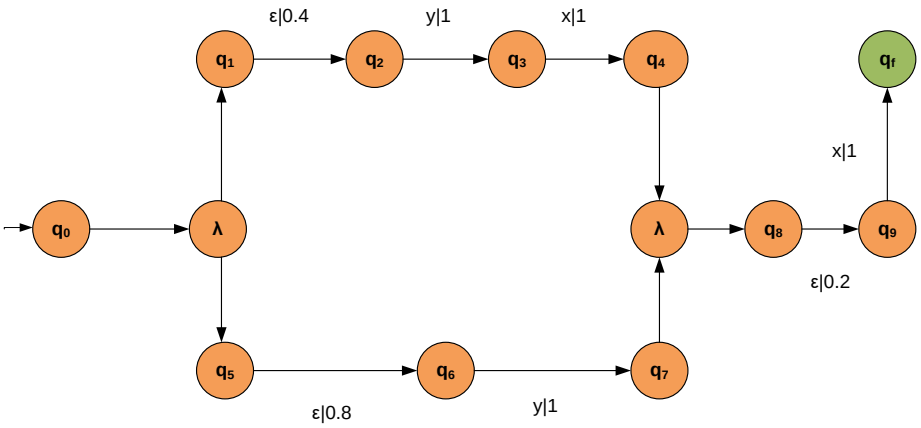

Fig. 3.1. Depicts the equivalent parallel fuzzy finite automata for $r = \left(0.4yx \;\&\; 0.8y\right)0.2x$

Consider £ be the product structure. Membership of string $w = yxyx$ can be determined using:

$$FL(M)w = \vee\sigma(q_0) \otimes \delta_{a_1}(q_0, q_1) \otimes \ldots \otimes \delta_{a_n}(q_{n-1}, q_n) \otimes \tau(q_n)$$

$$= \vee((1 \otimes 0.4 \otimes 1 \otimes 1 \otimes 0.8 \otimes 1 \otimes 0.2 \otimes 1 \otimes 1), (1 \otimes 0.8 \otimes 1 \otimes 0.4 \otimes 1 \otimes 1 \otimes 0.2 \otimes 1 \otimes 1), \ldots)$$

$$= 0.064$$

3.4 PROPOSED ALGORITHM

Given a parallel fuzzy regular expression r. Let Y be the set representing the symbols associated with different scalars appearing as the membership values in the parallel fuzzy

regular expression r. The parallel fuzzy regular expression r is converted into r_1, where scalars are replaced by their symbol from *Y*. This step resembles approaches used by various researchers (Refer to Stamenkovic and Ćirić [17], Mendivil and Garitagoitia [18], and Kuske [31] for details). Convert r_1 into $\bar{r}_1$ by assigning a position to each symbol of r_1 with respect to its appearance. Convert $\bar{r}_1$ into a parse tree such that $\phi, \varepsilon, a \in (\sum \bigcup Y)$ appear as a leaf node. Operators (union, concatenation, Kleene closure and shuffle) appear at an internal node, and inorder traversal of the parse tree is equal to $\bar{r}_1$.

Now, we will partition the parse tree into a sub-tree using modular decomposition. The modular decomposition technique was proposed by Yamamoto [45-46], and further used by Kumar and Verma [126] for improvement of the similar conversion. We will partition the parse tree into sub-trees such that:

1. The child of a shuffle operator or root of the parse tree will now be the root of the sub-tree.
2. The leaf node of a subtree can be any symbol from the extended alphabet or shuffle operator.
3. Each sub-tree does not contain a shuffle operator as the internal node.

We compute first, last and follow of each sub-tree using the well-known technique known as position automata [29-30]. The root of the parse tree appears in a subtree known as the root sub-tree. Nodes of a sub-tree represent a regular expression. The left and right sub-trees of the shuffle operator are considered as a divergence. Once the last position of the left and right sub-trees occurs, they will converge. Divergence and convergence in a module are represented by an edge $\lambda(i, j)$, where the i^{th} and j^{th} modules are children of the current module. We convert a non-deterministic automaton into an ε-fuzzy automaton using Mendivil and Garitagoitia's approach (refer [18] for more details). The algorithm *FPRE_FA* is proposed to obtain ε-fuzzy automata from fuzzy parallel regular expressions.

Algorithm 1: $FPRE\_FA(r, M)$

Input: Parallel fuzzy regular expression r.

Output: ε-free fuzzy automata M such that $FL(M) = FL(r)$.

1. Define a bijective function $F : L - \{0\} \to Y$ such that $\Sigma \bigcap Y = \phi$.

2. Convert r into r_1 by replacing each scalar a by $F(a)$.
3. Convert r_1 into $\overline{r_1}$ by assigning positions to the symbol of r_1.
4. Construct parse tree P_t for $\overline{r_1}$ such that
   a) $a \in \Sigma \cup Y$, ϕ, ε denotes leaf nodes of parse tree.
   b) Operators ., |, *and & appear as internal nodes.
   c) Inorder traversal of P_t is equal to $\overline{r_1}$.
5. Partition the parse tree P_t using modular decomposition.
6. **For** each subtree of P_t

   Compute $First, Last$ and $Follow$

   EndFor
7. Divergence and Convergence in a module are represented by an edge $\lambda(i, j)$ where i^{th} and j^{th} module are children of the current module.
8. Let $A_i(Q_1, \Sigma \cup Y, \sigma_1, \tau_1, \delta_1)$ and $A_j(Q_2, \Sigma \cup Y, \sigma_2 / 1, \tau_2, \delta_2)$ be two ε-free fuzzy automata representing the left (i^{th}) and right(j^{th}) sub-tree of a shuffle operator. Define $A_S(Q_1 \times Q_2, \Sigma \cup Y, (\sigma_1, \sigma_2), (\tau_1, \tau_2), \delta)$ such that $\delta((q_i, q_j), a) = (q_k, q_j) \cup (q_i, q_l)$ such that $(q_i, a, q_k) \in \delta_1 \wedge (q_j, a, q_l) \in \delta_2 \wedge a \in (\sum \cup Y \cup \varepsilon)$. Replace $\sigma(i, j)$ with respective A_S after adding ε-transition entering and leaving A_S.
9. **For** each transition (q_i, a, q_k) **/* Conversion of NFA to ε-Fuzzy Automata */**

   If $(a \in Y)$

   Convert transition (q_i, a, q_k) to $(q_i, \varepsilon \mid F^{-1}(a), q_k)$.

   Else

   Convert transition (q_i, a, q_k) to $(q_i, a \mid 1, q_k)$.

   EndIf

   EndFor

3.5 NUMERICAL EXAMPLE

In this section, we give an example to illustrate the algorithm 1 FPRE_FA.

Example 3.2: Given fuzzy parallel regular expression $r = (0.1x^*)((0.3yx \,\&\, 0.8y) + 0.2y)$.

Consider £ be the product structure.

$Y = \{\mu, \nu, o, \eta\}$,

$r_1 = (\mu x^*)((\nu yx \,\&\, oy) + \eta y)$, and

$\bar{r_1} = (\mu_1 x_2^*)((\nu_3 y_4 x_5 \,\&\, o_6 y_7) + \eta_8 y_9)$. Fig. 3.2 illustrates the parse tree for r_1.

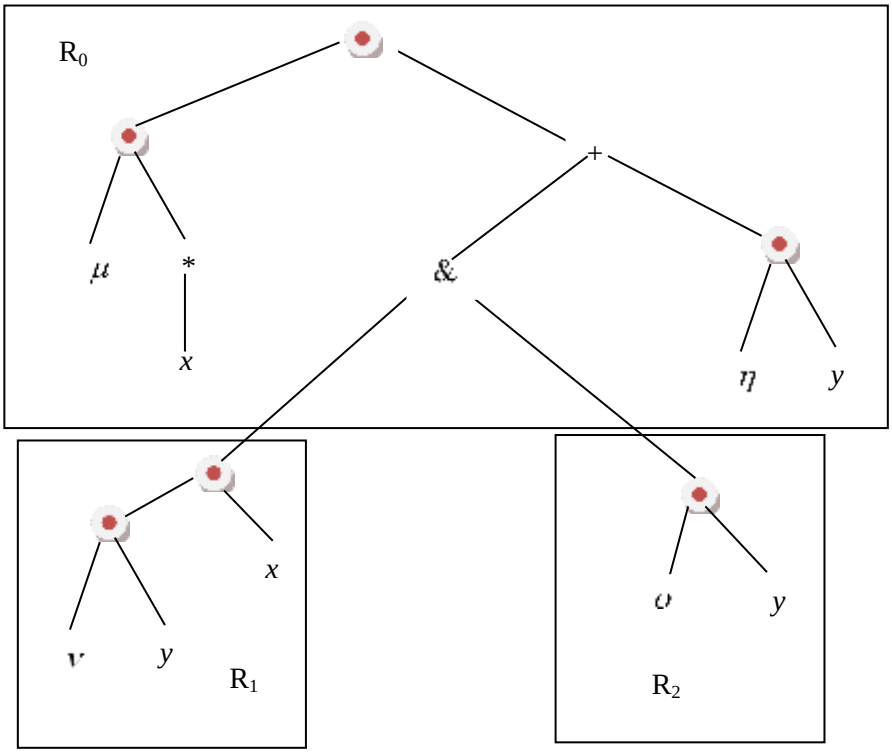

Fig. 3.2. Parse tree for fuzzy parallel regular expression $r_1 = (\mu x^*)((\nu yx \,\&\, oy) + \eta y)$.

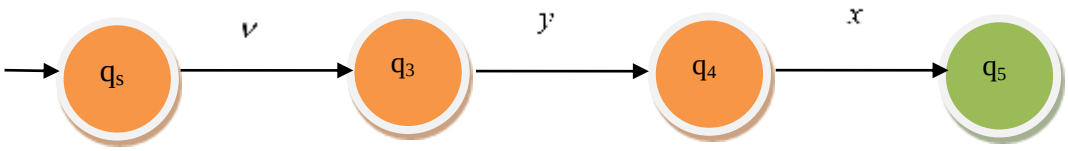

Fig. 3.3. NFA for module R_1

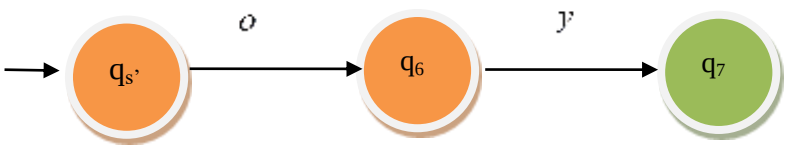

Fig. 3.4. NFA for module R_2

R_1 and R_2 modules are children of R_0. NFA for sub-tree R_1 and R_2 are shown in Fig. 3.3 and Fig. 3.4, respectively. NFA for module R_0 is represented in Fig. 3.5. Green color state depicts the final state.

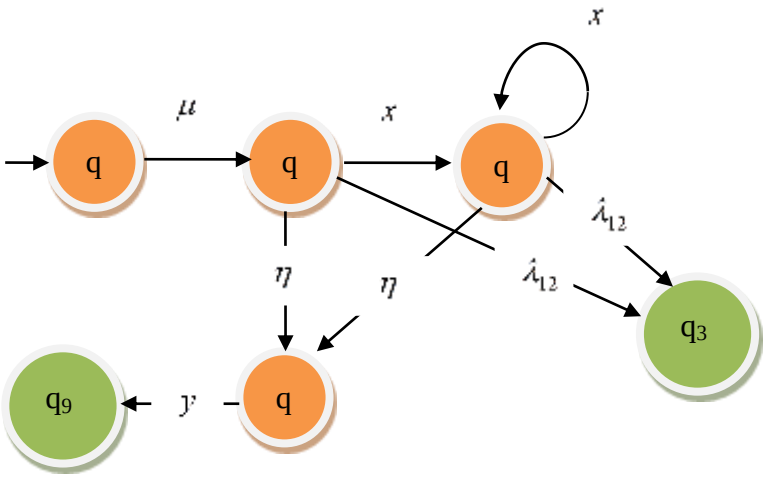

Fig. 3.5. NFA for module R_0, λ_{12} represent divergence to module R_1 and R_2

Fig. 3.6 represents ε-NFA for fuzzy parallel regular expression r. Divergence and Convergence represented by λ_{12} are replaced by ε and the Cartesian product of NFA for R_1 and R_2 gives us ε-NFA as shown in Fig.3.6.

ε-NFA of Fig. 3.6 can be converted into ε-fuzzy automata as shown in Fig. 3.7 using the following rules:

For each transition $(q_i, a, q_j) \wedge a \in Y$ convert it into $(q_i, \varepsilon \mid a, q_j)$.

For each transition $(q_i, a, q_j) \wedge a \in (\sum \cup \varepsilon)$ convert it into $(q_i, a \mid 1, q_j)$.

Mohri [79] proposed the generic epsilon removal for the weighted automata. Fuzzy automata are a special case of weighted automata. Using the ε-removal rules proposed by Mohri [79], we obtain the ε-free fuzzy automata as shown in Fig. 3.8.

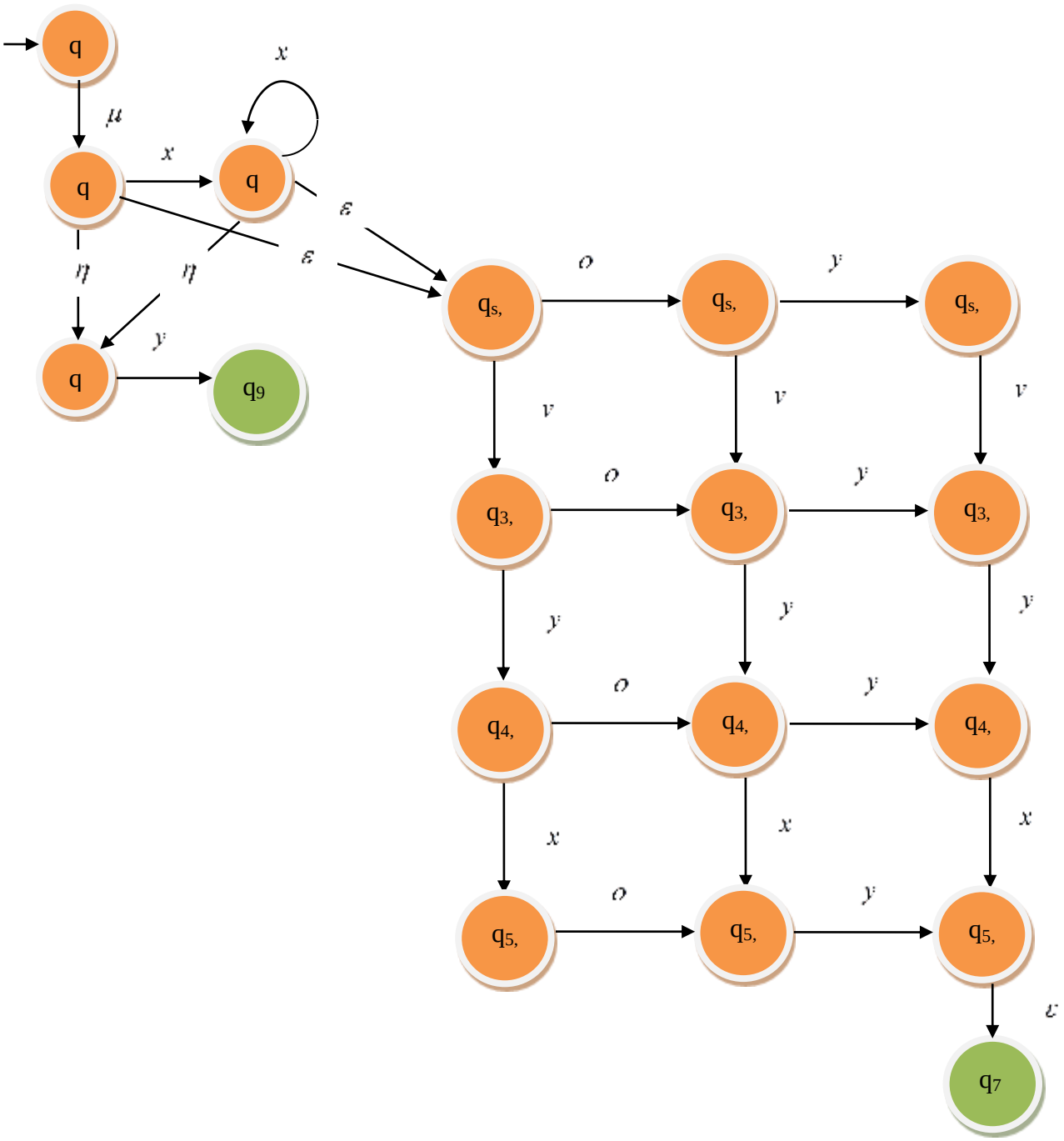

Fig. 3.6. ε – NFA for fuzzy parallel regular expression $r = (0.1x^{*})((0.3yx \,\&\, 0.8y) + 0.2y)$

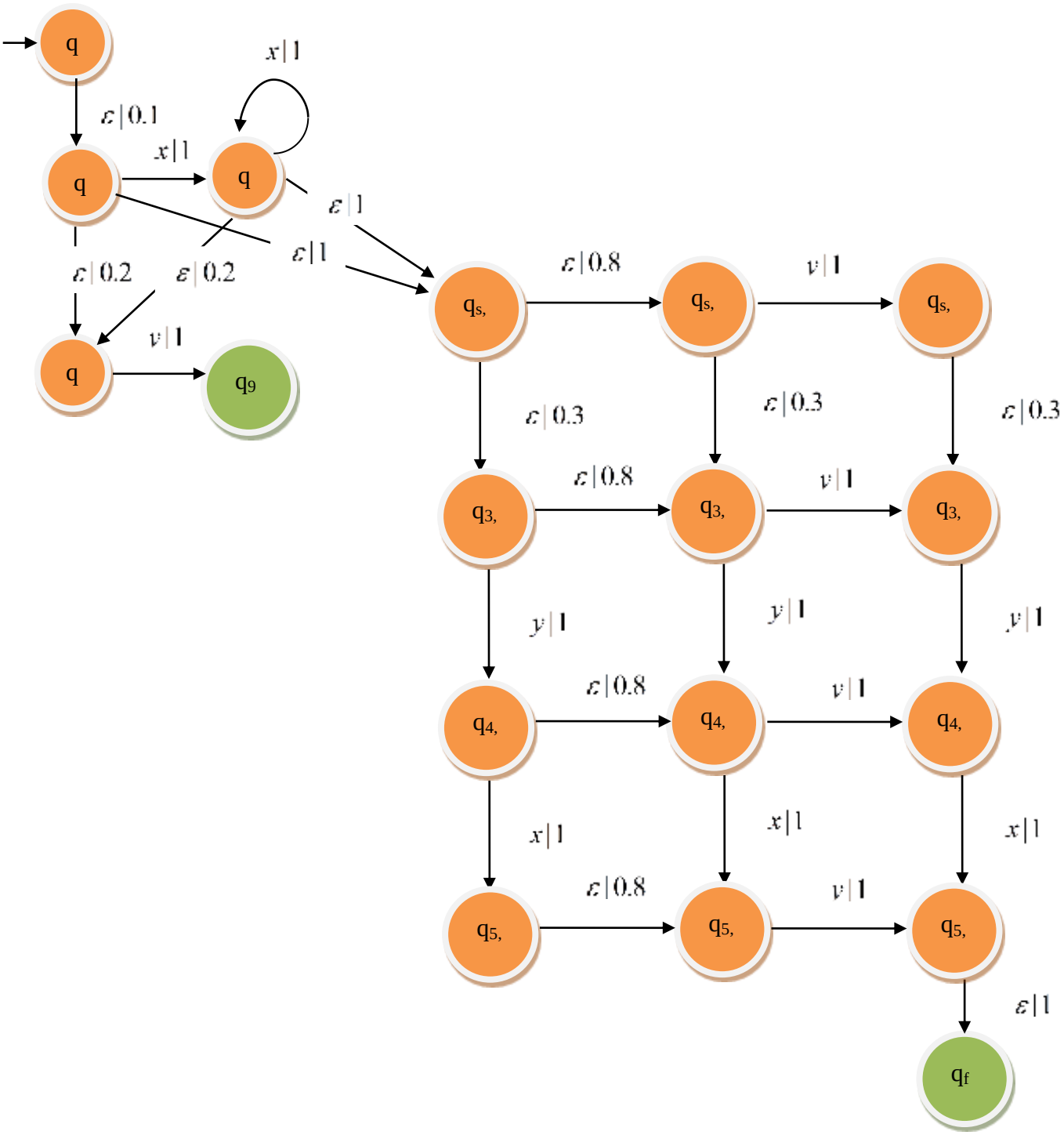

Fig. 3.7. ε – Fuzzy Automata for $r = (0.1x^*)((0.3yx \,\&\, 0.8y) + 0.2y)$

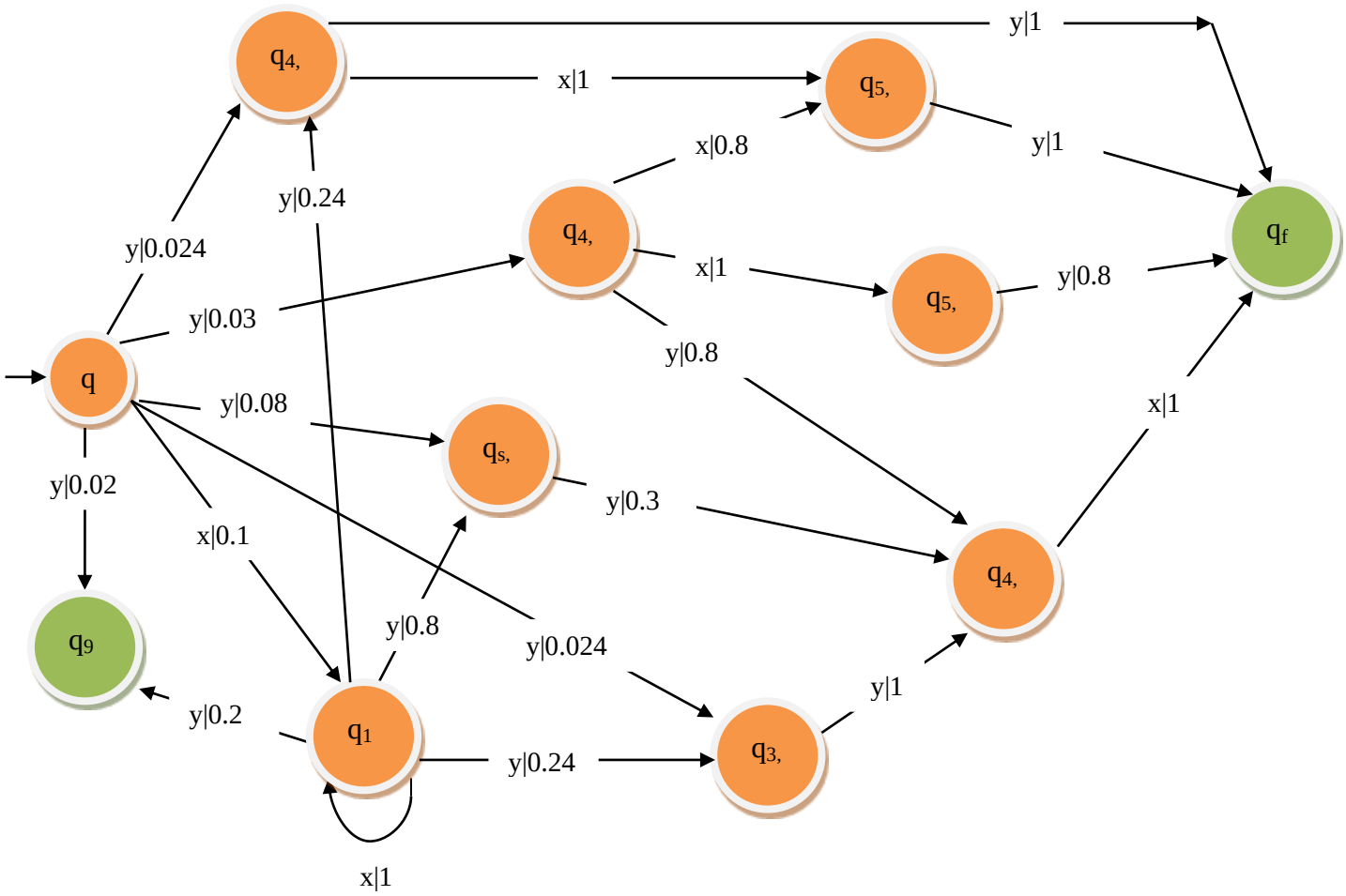

Fig. 3.8. $\varepsilon-$ free Fuzzy Automata for $r=(0.1x^{*})((0.3yx\,\&\,0.8y)+0.2y)$

3.6 RESULTS AND DISCUSSIONS

Theorem 3.1: Let r be a parallel fuzzy regular expression over integral ℓ-monoid. The number of states obtained in the ε-fuzzy automata using the *FPRE_FA* algorithm is equal to $(\sum_{I\in S}\Pi_{m\in I}(|m|+1))+2|s|$ in the worst case.

Proof. Let M represents a set of modules obtained by dividing r into the parse tree using the modular decomposition technique. Let $|m|$ be the total number of symbol appear in $m\in M$ from the extended alphabet. We have applied the position automata in each module, and the number of states is equal to $|m|+1$ for each module.

Let S be a set representing all independent subset of M [45-46]. The dependent modules are joined using the shuffle operator, and we have constructed fuzzy automata by taking the Cartesian products of the states appearing in the dependent modules. Two additional

states are added for each shuffle operator for representing divergence and convergence transitions in the parent module. For s shuffle operators in r, $2*s$ states are added for the convergence and divergence.

Hence, the number of states in the dependent modules is equal to $(\Pi_{I\in S, m\in I}(|m|+1))+2$.

Therefore, the number of states in ε -fuzzy automata is equal to $(\sum_{I\in S}\Pi_{m\in I}(|m|+1))+2^{*}s.$ □

The ε-fuzzy automata obtained using the *FPRE_FA* algorithm is converted into $\varepsilon-$ free fuzzy automata using the $\varepsilon-$ removal approach used by Mohri [79] for the weighted automata. All states having only incoming $\varepsilon-$ transitions are removed, which causes the number of states will be reduced.

3.7 CONCLUSION

In this chapter, we have introduced the concept of parallel fuzzy regular expressions and parallel fuzzy finite automata. Further, we have proposed an algorithm for the conversion of parallel fuzzy regular expressions to ε-free non-deterministic fuzzy automata. The proposed approach uses the concept of modular decomposition, position automata, extended alphabets and generic epsilon-removal algorithm.

Chapter-4

CONVERSION OF FUZZY AUTOMATA INTO REGULAR EXPRESSIONS USING TRANSITIVE CLOSURE

4.1 INTRODUCTION

Neumann [44] compared three approaches (transitive closure, state removal and Brzozowski algebraic) for the transformation of the finite automaton to a regular expression. Transitive closure is simple and easy to implement. State removal is difficult to implement but easy to recognize the regular expression manually. Brzozowski algebraic is easy to implement but best-suited for recursive languages. Singh [127] devised an algorithm for the conversion of deterministic finite automata to regular expressions using the bridge state concept. Thomason and Marinos [47] proposed a method for constructing deterministic fuzzy finite automata and its corresponding unambiguous grammar for the fuzzy regular expression. Cao and Ezawa [48] proposed the concept of non-deterministic fuzzy automata and recognize the corresponding fuzzy language. They proposed two variant of non-deterministic fuzzy automata with or without null moves. Further, they also proved that deterministic and non-deterministic fuzzy automata with or without null moves recognize the same class of fuzzy language i.e. all are equivalent in terms of powers.

Motivated from the existing literature for the transformation of finite automata to regular expression using approaches such as Arden's Theorem [40], transitive closure [41], Brzozowski algebraic [42], state removal [43], and in this chapter we propose an algorithm for the conversion of a fuzzy automaton into a fuzzy regular expression by extending the well-known concept of the transitive closure method [41]. Mainly transitive closure method is used for the conversion of deterministic finite automata into regular expressions. We support this discussion with the help of a numerical example. Further, we proved the equivalence of fuzzy automata and fuzzy regular expressions.

4.2 PROPOSED ALGORITHM

Results of this chapter have been published in Journal of Intelligent & Fuzzy Systems, Journal of Intelligent & Fuzzy Systems, 30(6), pp.3123-3129, 2016.

In this section, we use the transitive closure method for the conversion of fuzzy automata to fuzzy regular expressions. We consider the Kleene closure to have the highest priority in a fuzzy regular expression, followed by concatenation and then addition or scalar. The algorithm $FA\_FRE(M,r)$ converts a fuzzy automaton into fuzzy regular expression.

Algorithm 1: $FA\_FRE(M,r)$

Input: Fuzzy automaton M .

Output: Fuzzy regular expression r such that $L(r) = L(M)$.

1. ***For*** $i = 1$ to $i = n$ ***do***

   For $j = 1$ to $j = n$ ***do***

   If $(i == j)$ then

   $R[i, j, 0] = R[i, j, 0] + 1.0\varepsilon$

   Fi

   do $e \in E$ such that $(i, j) = e$

   // consider for input alphabet a there exists an edge from state i to state j with fuzzy membership λ_{ij}

   $R[i, j, 0] = R[i, j, 0] + \lambda_{ij}.a$

   od

   od

   od

2. ***For*** $k = 1$ to $k = n$ ***do***

   For $i = 1$ to $i = n$ ***do***

   For $j = 1$ to $j = n$ ***do***

   $R[i, j, k] = R[i, j, k-1] + R[i, k, k-1] \otimes (R[k, k, k-1] \otimes)^* R[k, j, k-1]$

   od

   od

   od

3. $R = \phi$

4. ***For*** $\exists i \in F$ ***do***

   $R = R \vee (\sigma(1) \otimes R_{1i}^n \otimes \tau(q_i))$

   od

4.3 NUMERICAL EXAMPLE

In this section, the modified transitive closure method has been successfully applied to construct a fuzzy regular expression from the fuzzy automaton. This example is taken from [17-18].

Example 4.1: Given the fuzzy automaton as shown in Fig. 4.1.

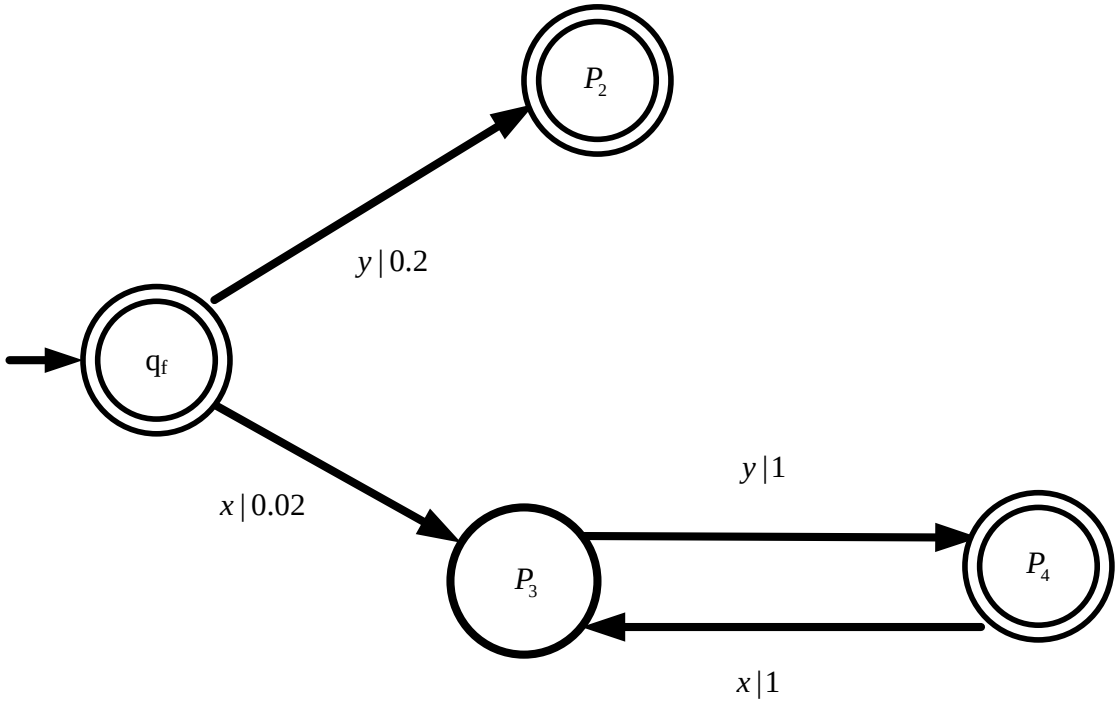

Fig. 4.1: Fuzzy Automaton $(\{P_1, P_2, P_3, P_4\}, \{x, y\}, \delta, \{P_1 / 1.0\}, \{P_1 / 0.2, P_3 / 1.0, P_4 / 1.0\})$

The final state with their membership values are $\{(P_1 / 0.2), (P_3 / 1), (P_4 / 1)\}$.

When $k = 0$ following are the initial fuzzy regular expressions:

$R_{11}^0 = 1.0\varepsilon \quad R_{21}^0 = \phi \quad R_{31}^0 = \phi \quad R_{41}^0 = \phi$

$R_{12}^0 = 0.2y \quad R_{22}^0 = 1.0\varepsilon \quad R_{32}^0 = \phi \quad R_{42}^0 = \phi$

$R_{13}^0 = 0.02x \quad R_{23}^0 = \phi \quad R_{33}^0 = 1.0\varepsilon \quad R_{43}^0 = 1.0x$

$R_{14}^0 = \phi \quad R_{24}^0 = \phi \quad R_{34}^0 = 1.0y \quad R_{44}^0 = 1.0\varepsilon$

$$R_{ij}^k = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)} \; k = 1 \qquad (1)$$

$$R_{11}^1 = R_{11}^0 + R_{11}^0 (R_{11}^0)^* R_{11}^0 = 1.0\varepsilon + 1.0\varepsilon (1.0\varepsilon)^* 1.0\varepsilon = 1.0\varepsilon \quad (2)$$

$$R_{12}^1 = R_{12}^0 + R_{11}^0 (R_{11}^0)^* R_{12}^0 = 0.2y + 1.0\varepsilon (1.0\varepsilon)^* 0.2y = 0.2y \quad (3)$$

$$R_{13}^1 = R_{13}^0 + R_{11}^0 (R_{11}^0)^* R_{13}^0 = 0.02x + 1.0\varepsilon (1.0\varepsilon)^* 0.02x = 0.02x \quad (4)$$

$$R_{14}^1 = R_{14}^0 + R_{11}^0 (R_{11}^0)^* R_{14}^0 = \phi \quad (5)$$

$$R_{21}^1 = R_{21}^0 + R_{21}^0 (R_{11}^0)^* R_{11}^0 = \phi \quad (6)$$

$$R_{22}^1 = R_{22}^0 + R_{21}^0 (R_{11}^0)^* R_{12}^0 = 1.0\varepsilon + \phi = 1.0\varepsilon \quad (7)$$

$$R_{23}^1 = R_{23}^0 + R_{21}^0 (R_{11}^0)^* R_{13}^0 = \phi + \phi = \phi \quad (8)$$

$$R_{24}^1 = R_{24}^0 + R_{21}^0 (R_{11}^0)^* R_{14}^0 = \phi + \phi = \phi \quad (9)$$

$$R_{3j}^1 = R_{3j}^0 + R_{31}^0 (R_{11}^0)^* R_{1j}^0 \quad (10)$$

$$R_{31}^1 = R_{31}^0 + R_{31}^0 (R_{11}^0)^* R_{11}^0 = \phi + \phi = \phi \quad (11)$$

$$R_{32}^1 = R_{32}^0 + R_{31}^0 (R_{11}^0)^* R_{12}^0 = \phi + \phi = \phi \quad (12)$$

$$R_{33}^1 = R_{33}^0 + R_{31}^0 (R_{11}^0)^* R_{13}^0 = 1.0\varepsilon \quad (13)$$

$$R_{34}^1 = R_{34}^0 + R_{31}^0 (R_{11}^0)^* R_{14}^0 = y + \phi = y \quad (14)$$

$$R_{4j}^1 = R_{4j}^0 + R_{41}^0 (R_{11}^0)^* R_{1j}^0 = R_{4j}^0 \quad (15)$$

$$R_{41}^1 = \phi \;\; R_{42}^1 = \phi \;\; R_{43}^1 = 1.0x \;\; R_{44}^1 = 1.0\varepsilon \quad (16)$$

$$R_{ij}^k = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)} \;\; k = 2 \quad (17)$$

$$R_{1j}^2 = R_{1j}^1 + R_{12}^1 (R_{22}^1)^* R_{2j}^1 \quad (18)$$

$$R_{11}^2 = R_{11}^1 + R_{12}^1 (R_{22}^1)^* R_{21}^1 = 1.0\varepsilon + \phi = 1.0\varepsilon \quad (19)$$

$$R_{12}^2 = R_{12}^1 + R_{12}^1 (R_{22}^1)^* R_{22}^1 = 0.2y + 0.2y (1.0\varepsilon)^* 1.0\varepsilon = 0.2y \quad (20)$$

$$R_{13}^2 = R_{13}^1 + R_{12}^1 (R_{22}^1)^* R_{23}^1 = 0.02x + \phi = 0.02x \quad (21)$$

$$R_{14}^2 = R_{14}^1 + R_{12}^1 (R_{22}^1)^* R_{24}^1 = \phi + 0.2y(1.0\varepsilon)^* \phi = \phi \quad (22)$$

$$R_{2j}^2 = R_{2j}^1 + R_{22}^1 (R_{22}^1)^* R_{2j}^1 \quad (23)$$

$$R_{21}^2 = R_{21}^1 + R_{22}^1 (R_{22}^1)^* R_{21}^1 = \phi + 1.0\varepsilon(1.0\varepsilon)^* \phi = \phi \quad (24)$$

$$R_{22}^2 = R_{22}^1 + R_{22}^1 (R_{22}^1)^* R_{22}^1 = 1.0\varepsilon + 1.0\varepsilon(1.0\varepsilon)^* 1.0\varepsilon = 1.0\varepsilon \quad (25)$$

$$R_{23}^2 = R_{23}^1 + R_{22}^1 (R_{22}^1)^* R_{23}^1 = \phi + 1.0\varepsilon(1.0\varepsilon)^* \phi = \phi \quad (26)$$

$$R_{24}^2 = R_{24}^1 + R_{22}^1 (R_{22}^1)^* R_{24}^1 = \phi + \phi = \phi \quad (27)$$

$$R_{3j}^2 = R_{3j}^1 + R_{33}^1 (R_{33}^1)^* R_{3j}^1 \quad (28)$$

$$R_{3j}^2 = R_{3j}^1 + R_{33}^1 (R_{33}^1)^* R_{3j}^1 = R_{3j}^1 + \varepsilon R_{3j}^1 \quad (29)$$

$$R_{31}^2 = \phi + \phi = \phi \quad (30)$$

$$R_{32}^2 = R_{32}^1 + 1.0\varepsilon R_{32}^1 = \phi \quad (31)$$

$$R_{33}^2 = R_{33}^1 + 1.0\varepsilon R_{33}^1 = 1.0\varepsilon \quad (32)$$

$$R_{34}^2 = R_{34}^1 + 1.0\varepsilon R_{34}^1 = 1.0y + 1.0\varepsilon y = 1.0y \quad (33)$$

$$R_{ij}^k = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)} \quad (34)$$

$$R_{4j}^2 = R_{4j}^1 + R_{4k}^1 (R_{kk}^1)^* R_{kj}^1 \quad (35)$$

$$R_{41}^2 = R_{41}^1 + R_{42}^1 (R_{22}^1)^* R_{21}^1 = \phi \quad (36)$$

$$R_{42}^2 = R_{42}^1 + R_{42}^1 (R_{22}^1)^* R_{22}^1 = \phi \quad (37)$$

$$R_{43}^2 = R_{43}^1 + R_{42}^1 (R_{22}^1)^* R_{23}^1 = 1.0x + \phi = 1.0x \quad (38)$$

$$R_{44}^2 = R_{44}^1 + R_{42}^1 (R_{22}^1)^* R_{24}^1 = 1.0\varepsilon + \phi = 1.0\varepsilon \quad (39)$$

$$R_{ij}^{k} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^{*} R_{kj}^{(k-1)} \; k = 3 \tag{40}$$

$$R_{1j}^{3} = R_{1j}^{2} + R_{13}^{2}(R_{33}^{2})^{*} R_{3j}^{2} \tag{41}$$

$$R_{11}^{3} = R_{11}^{2} + R_{13}^{2}(\varepsilon)^{*} R_{31}^{2} = 1.0\varepsilon + \phi = 1.0\varepsilon \tag{42}$$

$$R_{12}^{3} = R_{12}^{2} + R_{13}^{2}(\varepsilon)^{*} R_{32}^{2} = 0.2y \tag{43}$$

$$R_{13}^{3} = R_{13}^{2} + R_{13}^{2}(\varepsilon)^{*} R_{33}^{2} = 0.02x + 0.02x(1.0\varepsilon)^{*} 1.0\varepsilon = 0.02x \tag{44}$$

$$R_{14}^{3} = R_{14}^{2} + R_{13}^{2}(\varepsilon)^{*} R_{34}^{2} = \phi + (0.02x)1.0y = (0.02x)y \tag{45}$$

$$R_{2j}^{3} = R_{2j}^{2} + R_{23}^{2}(R_{33}^{2})^{*} R_{3j}^{2} \tag{46}$$

$$R_{21}^{3} = R_{21}^{2} + R_{23}^{2}(R_{33}^{2})^{*} R_{21}^{2} = \phi \tag{47}$$

$$R_{22}^{3} = R_{22}^{2} + R_{23}^{2}(R_{33}^{2})^{*} R_{32}^{2} = 1.0\varepsilon + \phi = 1.0\varepsilon \tag{48}$$

$$R_{23}^{3} = R_{23}^{2} + R_{23}^{2}(R_{33}^{2})^{*} R_{33}^{2} = \phi \tag{49}$$

$$R_{24}^{3} = R_{24}^{2} + R_{23}^{2}(R_{33}^{2})^{*} R_{34}^{2} = \phi \tag{50}$$

$$R_{3j}^{3} = R_{3j}^{2} + R_{33}^{2}(R_{33}^{2})^{*} R_{3j}^{2} = R_{3j}^{2} + R_{3j}^{2} \tag{51}$$

$$R_{31}^{3} = \phi \; R_{32}^{3} = \phi \; R_{33}^{3} = \varepsilon \; R_{34}^{3} = 1.0y \tag{52}$$

$$R_{4j}^{3} = R_{4j}^{2} + R_{43}^{2}(R_{33}^{2})^{*} R_{3j}^{2} = R_{4j}^{2} + xR_{3j}^{2} \tag{53}$$

$$R_{41}^{3} = \phi + 1.0x\phi = \phi \tag{54}$$

$$R_{42}^{3} = \phi + 1.0x\phi = \phi \tag{55}$$

$$R_{43}^{3} = R_{43}^{2} + R_{43}^{2}(R_{33}^{2})^{*} R_{33}^{2} = 1.0x + 1.0x = 1.0x \tag{56}$$

$$R_{44}^{3} = 1.0\varepsilon + 1.0x1.0y = 1.0\varepsilon + xy \; k = 4 \tag{57}$$

$$R_{1j}^{4} = R_{1j}^{3} + R_{14}^{3}(R_{44}^{3})^{*} R_{4j}^{3} \tag{58}$$

$$R_{11}^4 = \varepsilon + \phi = 1.0\varepsilon \tag{59}$$

$$R_{12}^4 = R_{12}^3 + R_{14}^3 (R_{44}^3)^* R_{42}^3 = 0.2y + (0.02x)1.0y(1.0\varepsilon + 1.0x1.0y)^* \phi = 0.2y \tag{60}$$

$$R_{13}^4 = R_{13}^3 + (0.02x)1.0y(1.0\varepsilon + 1.0x1.0y)^* R_{43}^3 = 0.02x + (0.02x)1.0y(1.0\varepsilon + 1.0x1.0y)^* 1.0x$$

$$= 0.02x + (0.02x)y(xy)^* x = 0.02x(1.0\varepsilon + y(xy)^* x) = (0.02x)(yx)^* \tag{61}$$

$$R_{14}^4 = R_{14}^3 + R_{14}^3 (R_{44}^3)^* R_{44}^3$$

$$= (0.02x)1.0y + (0.02x)1.0y(1.0\varepsilon + 1.0x1.0y)^* (1.0\varepsilon + 1.0x1.0y) \tag{62}$$

$$R_{2j}^4 = R_{2j}^3 + R_{24}^3 (R_{44}^3)^* R_{4j}^3 \tag{63}$$

$$R_{21}^4 = R_{21}^3 + \phi = \phi \tag{64}$$

$$R_{22}^4 = 1.0\varepsilon \tag{65}$$

$$R_{23}^4 = \phi \tag{66}$$

$$R_{24}^4 = \phi \tag{67}$$

$$R_{3j}^4 = R_{3j}^3 + R_{34}^3 (R_{44}^3)^* R_{4j}^3 = R_{3j}^3 + y(\varepsilon + xy)^* R_{4j}^3 \tag{68}$$

$$R_{31}^4 = \phi + 1.0y(1.0\varepsilon + 1.0xy)^* \phi = \phi \tag{69}$$

$$R_{32}^4 = \phi + \phi = \phi \tag{70}$$

$$R_{33}^4 = 1.0\varepsilon + 1.0y(1.0\varepsilon + 1.0xy)^* x = 1.0\varepsilon + y(xy)^* x = (yx)^* \tag{71}$$

$$R_{34}^4 = R_{34}^3 + R_{34}^3 (R_{44}^3)^* R_{44}^3 = 1.0y + 1.0y(1.0\varepsilon + 1.0xy)^* (1.0\varepsilon + 1.0xy)$$

$$= 1.0y(1.0\varepsilon + (1.0\varepsilon + 1.0xy)^* (1.0\varepsilon + 1.0xy)) = y(\varepsilon + xy)^* = y(xy)^* \tag{72}$$

$$R_{4j}^4 = R_{4j}^3 + R_{44}^3 (R_{44}^3)^* R_{4j}^3 = R_{4j}^3 + (\varepsilon + xy)(xy)^* R_{4j}^3 \tag{73}$$

$$R_{41}^4 = R_{41}^3 + R_{44}^3 (R_{44}^3)^* R_{41}^3 = R_{4j}^3 + (1.0\varepsilon + 1.0xy)(1.0xy)^* R_{4j}^3 = \phi \tag{74}$$

$$R_{42}^4 = R_{42}^3 + R_{44}^3 (R_{44}^3)^* R_{42}^3 = \phi \quad (75)$$

$$R_{43}^4 = R_{43}^3 + R_{44}^3 (R_{44}^3)^* R_{43}^3 = 1.0x + (1.0\varepsilon + 1.0xy)(1.0xy)^* 1.0x \quad (76)$$

$$R_{44}^4 = R_{44}^3 + R_{44}^3 (R_{44}^3)^* R_{44}^3 = (1.0\varepsilon + 1.0xy) + (1.0\varepsilon + 1.0xy)(1.0xy)^* (1.0\varepsilon + 1.0xy) \quad (77)$$

Final states are $\{(P_1 / 0.2), (P_3 / 1), (P_4 / 1)\}$

$R_{11}^4 = \{\varepsilon\}$ and final state is having membership 0.2

Hence $R_{11}^4 = \{1.0 \otimes 0.2\varepsilon\} = 0.2\varepsilon \quad (78)$

$R_{12}^4 = \{0.2y\}$ and final state is having membership 1

Hence $R_{12}^4 = \{1.0 \otimes 0.2y \otimes 1.0\} \quad (79)$

$$R_{14}^4 = 1.0 \otimes 1.0 \otimes (0.02x)y + (0.02x)y(\varepsilon + xy)^* (\varepsilon + xy)$$

$$= (0.02x)y + (0.02x)y(\varepsilon + xy)^* (\varepsilon + xy)$$

$$= (0.02x)y(\varepsilon + (\varepsilon + xy)^* (\varepsilon + xy)) = (0.02x)y(\varepsilon + (\varepsilon + xy)^* (\varepsilon + xy))$$

$$= (0.02x)y(\varepsilon + xy)^+ = (0.02x)y(xy)^* \quad (80)$$

Final regular expression equivalent to fuzzy automaton

$$= R_{11}^4 + R_{12}^4 + R_{14}^4 = 0.2 + 0.2y + (0.02x)y(xy)^*$$

$$= 0.2(\varepsilon + (0.1x)y(xy)^*) + 0.2y = 0.2((0.1)(xy)^*) + 0.2y$$

$$= 0.2((0.1)(xy)^*) + y) \quad (81)$$

4.4 RESULTS AND DISCUSSIONS

Theorem 4.1: Given a fuzzy automaton $M = (Q, X, \delta, \sigma, \tau)$, then there exists a fuzzy regular expression r obtained from M using the algorithm *FA_FRE* (M, r) such that $L(M) = L(r)$.

Proof: Let $|Q| = n$. Rename the states name of M to first n positive numbers $\{1, 2, ..., n\}$. This theorem is established by mathematical induction.

R_{ij}^{k} denotes the fuzzy regular expressions using the path from state i to state j such that no intermediate node in the path is greater than k.

Basis step: Consider that $k = 0$ means no intermediate states are presenting.

For $\exists i, j \in n$, Initialize $R_{ij}^{0} = \phi$

For each transition from i to state j, the following three cases can occur:

Case 1: If there exists only one symbol a from the state i to the state j with membership value λ as delineated in Fig.4.2, then

$$R_{ij}^{0} = R_{ij}^{0} + \lambda a \tag{82}$$

Case 2: If there are symbols $a_1, a_2, ..., a_l$ with their respective membership from state i to the state j with membership values $\lambda_1, \lambda_2, ..., \lambda_l$ as delineated in Fig. 4.3, then

$$R_{ij}^{0} = R_{ij}^{0} + \lambda_1 a_1 + \lambda_1 a_2 + ... + \lambda_l a_l \tag{83}$$

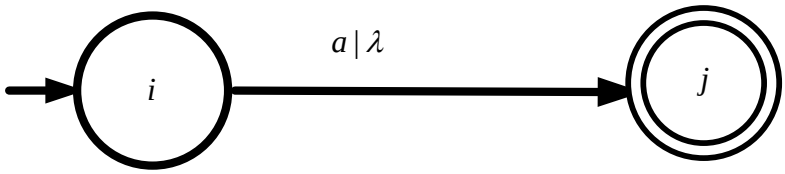

Fig. 4.2: Transition from state i to state j with the symbol a

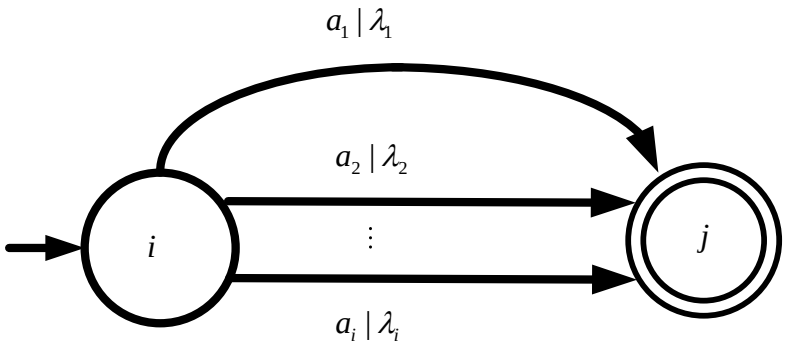

Fig. 4.3: Transition from state i to state j with symbols $a_1, a_2, ..., a_l$

Case 3: If $i = j$

$$R_{ij}^{0} = R_{ij}^{0} + \lambda\varepsilon \text{ where } \lambda = 1 \tag{84}$$

Induction step: Now, we consider the path from state i to state j with intermediate state is not less than k. The following two cases can occur:

Case 1: If there exists no new path from state i to state j using intermediate state k [40], then

$$R_{ij}^{k} = R_{ij}^{k-1} \tag{85}$$

Case 2: This case represents the intermediate state k case, whenever we are referring path from state i to the state j. The fuzzy regular expression R_{ij}^{k} (as shown in Fig. 4.4) through intermediate state k can be found using

$$R_{ij}^{k} = R_{ik}^{k-1} \otimes (R_{kk}^{k-1}\otimes)^{*} R_{kj}^{k-1} \tag{86}$$

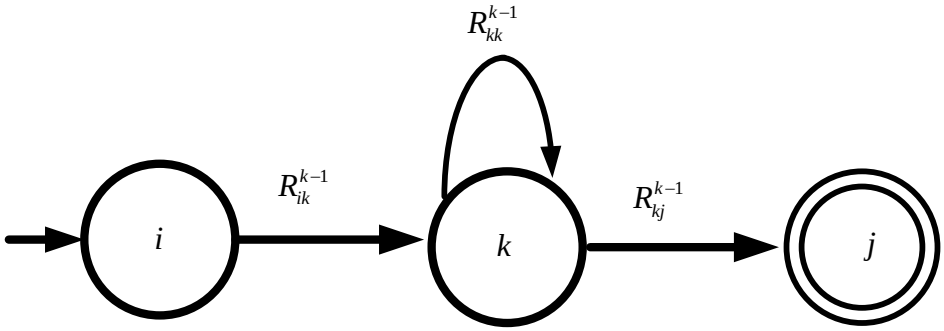

Fig.4.4: Computation of fuzzy regular expression R_{ij}^{k}

- The expression R_{ik}^{k-1} represents the fuzzy regular expression from state i to the state k using intermediate state from $\{1, 2, ..., k-1\}$.
- The expression $(R_{kk}^{k-1})^{*}$ represents the fuzzy regular expression from state k to the state k using intermediate state from $\{1, 2, ..., k-1\}$.
- The expression R_{kj}^{k-1} represents the fuzzy regular expression from state k to the state j using intermediate state from $\{1, 2, ..., k-1\}$.

Hence, $R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1} \otimes (R_{kk}^{k-1}\otimes)^{*} R_{kj}^{k-1}$ (87)

Consider the starting state q_1 is having the membership value and final state $F = \{q_l, q_m, ...\}$. Consider the fuzzy membership value of the initial state be $\sigma(1)$.

The final fuzzy regular expression equivalent to the fuzzy automaton is as follows:

$$R = \bigcup_{i \in F} (\sigma(1) \otimes R_{1i}^{n} \otimes \tau(q_i)) \tag{88}$$ □

The proof of Theorem 4.1 is an enhancement of the transitive closure method [40] for the conversion of deterministic finite automata to regular expressions with the inclusion of the fuzziness concept. Further, we need to simplify the obtained intermediate fuzzy regular expressions at each iteration in order to obtain smaller fuzzy regular expression.

4.5 CONCLUSION

From an experimental perspective, we have proposed an algorithm for the conversion of fuzzy automata into fuzzy regular expressions. This algorithm is founded on the concept of the transitive closure. Furthermore, we have proven that there exists a fuzzy regular expression r corresponding to the fuzzy automaton M such that $L(r) = L(M)$. This method is computationally attractive, as demonstrated through the illustrative example.

Chapter-5

FUZZY LINEAR EXPRESSION AND TWO SIDED PARTIAL DERIVATIVES

5.1 INTRODUCTION

Sempere [74] introduced the concept of linear expression for representing context-free linear language. Further, he described an approach for the conversion of linear expression to/from linear grammar. In this chapter, we introduced the concept of fuzzy linear expression to represent fuzzy linear context-free languages. Garhwal and Jiwari [128] proposed the concept of fuzziness in parallel regular expression. Kalra and Kumar [129] introduced the concept of fuzziness in the regulated grammar. Bio-molecular structures (Inverted repeat, Hairpin structure, Stem and Loop) are represented using linear expressions. An algorithm is proposed for the conversion of a fuzzy linear grammar to its normal form. Further, we have extended the conversion of linear grammars to linear expressions by inclusion of fuzziness.

Partial derivatives are used in classical automata theory (for details, refer to [42, 49-56]) for the conversion of regular expressions to finite automata [56], regular expressions to tree automata [50], and ω-regular expressions to Büchi automata [49]. Allauzen and Mohri [80] described a unified approach for the conversion of weighted regular expressions to weighted automata using Glushkov, Follow, and Antimirov Automata. Armed with such a vision, the similar concept of the partial derivative will be utilized in this work to obtain fuzzy automata from fuzzy regular expressions. Use of Antimirov's partial derivatives [56] in classical automata theory is an important issue because it generates a smaller ε-free non-deterministic finite automaton with a maximum of $(m+1)$ states. This motivates the concept of partial derivatives to be addressed in fuzzy regular expression. In the last Section, we described a new variant of Antimirov's partial derivatives for the conversion of fuzzy regular expressions to ε-free fuzzy non-deterministic finite automata. The resulting automaton is an improvement over the

[2]Results of this chapter have been accepted in the Conference "International Conference on High Performance Computing & Simulation (HPCS 2017), Genoa, Italy".

existing approaches, and the resulting automaton is ε-free. Further, we have proved that non-deterministic fuzzy automaton M is obtained from a fuzzy regular expression r such that $LR(M) = LR(r)$.

5.2 ANTIMIROV'S PARTIAL DERIVATIVE FOR NFA CONSTRUCTION

Consider regular expression r represents the language L over input alphabet Σ. Antimirov [56] proposed the following conditions of the partial derivatives given $\Sigma = \{a, b\}$:

$$\partial_a(\phi) = \phi, \quad (1)$$

$$\partial_a(\varepsilon) = \phi, \quad (2)$$

$$\partial_a(b) = \phi, \quad (3)$$

$$\partial_a(a) = \varepsilon, \quad (4)$$

$$\partial_a(r_1 \bigcup r_2) = \partial_a(r_1) \bigcup \partial_a(r_2), \quad (5)$$

$$\partial_a(r_1^*) = \partial_a(r_1)(r_1^*), \quad (6)$$

If $\varepsilon \in L(r_1)$ then

$$\partial_a(r_1 r_2) = \partial_a(r_1) r_2 \bigcup \partial_a(r_2), \quad (7,a)$$

Else

$$\partial_a(r_1 r_2) = \partial_a(r_1) r_2, \quad (7,b)$$

End If

Starting state is defined by $q_0 = r$, (8)

Final state F of the *NFA* is defined by $F = \{q \mid q \in Q \wedge \varepsilon \in L(q)\}$. (9)

Example 5.1: Consider the regular expression $r = b^*ba(a+b)^*$ over $\sum = \{a, b\}$. The *NFA* construction starts with $q_0 = b^*ba(a+b)^*$ and proceeds as follows:

1. Compute $\partial_a(b^*ba(a+b)^*) = \phi$
2. Compute $\partial_b(b^*ba(a+b)^*) = (b^*ba(a+b)^*) \bigcup a(a+b)^* = q_0 \bigcup q_1$, the new state denoted by q_1.
3. Compute $\partial_a(a(a+b)^*) = (a+b)^* = q_2$.

4. Compute $\partial_b(a(a+b)^*) = \phi$.
5. Compute $\partial_a((a+b)^*) = (a+b)^* = q_2$.
6. Compute $\partial_b((a+b)^*) = (a+b)^* = q_2$.

Since $\varepsilon \in q_2$, then q_2 is a final state. Fig. 5.1 depicts the *NFA* corresponding to $r = b^*ba(a+b)^*$.

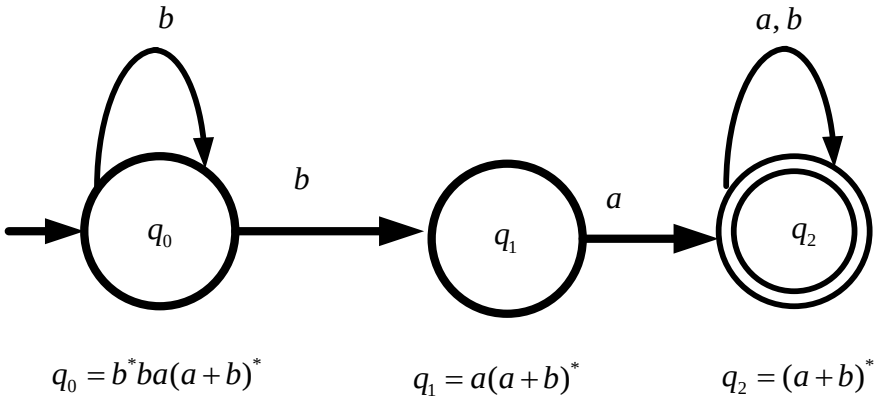

Fig. 5.1. The *NFA* for $r = b^*ba(a+b)^*$ using Antimirov's partial derivatives.

5.3 TWO SIDED DERIVATIVES FOR REGULAR EXPRESSIONS

Champarnaud et al. [76] introduced the concept of two sided derivatives for regular expressions and applied it for hairpin structures. Left partial derivatives are similar as Antimirov's partial derivatives defined in Section 5.2. Right partial derivatives [76] are defined using the following rules:

$$(a)\frac{\partial}{\partial_a} = \varepsilon \tag{10}$$

$$(a)\frac{\partial}{\partial_b} = \varepsilon \tag{11}$$

$$(r_1 + r_2)\frac{\partial}{\partial_a} = (r_1)\frac{\partial}{\partial_a} \bigcup (r_2)\frac{\partial}{\partial_a} \tag{12}$$

$$(r_1^*)\frac{\partial}{\partial_a} = r_1^*(r_1)\frac{\partial}{\partial_a} \tag{13}$$

$$(r_1 r_2)\frac{\partial}{\partial_a} = \begin{cases} r_1 . (r_2)\frac{\partial}{\partial_a} \cup (r_1)\frac{\partial}{\partial_a} & \text{if } \varepsilon \in L(r_2) \\ r_1 . (r_2)\frac{\partial}{\partial_a} & \text{otherwise} \end{cases} \quad (14)$$

$$(r_1)\frac{\partial}{\partial_{a.w}} = \left((r_1)\frac{\partial}{\partial_a}\right)\frac{\partial}{\partial_w} \quad (15)$$

$$(r_1)\frac{\partial}{\partial_\varepsilon} = \{r_1\} \quad (16)$$

Example 5.2: Consider the regular expression $r = b^*ba(a+b)^*$ over $\sum = \{a,b\}$. The *NFA* construction starts with $q_0 = b^*ba(a+b)^*$ and proceeds as follows using right partial derivative:

$$\left(b^*ba(a+b)^*\right)\frac{\partial}{\partial_a} = \left(b^*ba(a+b)^*\right)(a+b)\frac{\partial}{\partial_a} \cup (b^*b)(a)\frac{\partial}{\partial_a}$$

$$= q_0 \cup (b^*b) = q_0 \cup q_1 \quad (17)$$

$$\left(b^*ba(a+b)^*\right)\frac{\partial}{\partial_b} = \left(b^*ba(a+b)^*\right)(a+b)\frac{\partial}{\partial_b} \cup (b^*b)(a)\frac{\partial}{\partial_b}$$

$$= q_0 \cup \phi \quad (18)$$

$$\left(b^*b\right)\frac{\partial}{\partial_a} = b^*(b)\frac{\partial}{\partial_a} = \phi \quad (19)$$

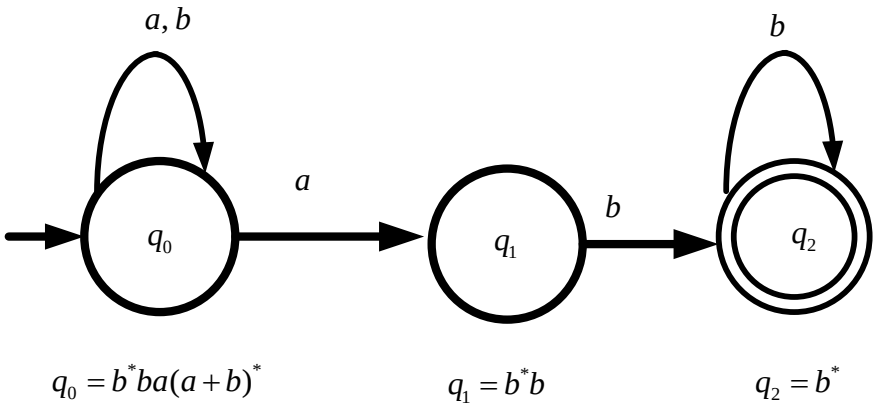

Fig. 5.2. The *NFA* for $r = b^*ba(a+b)^*$ using right-derivatives

$$\left(b^*b\right)\frac{\partial}{\partial_b} = b^*(b)\frac{\partial}{\partial_b} = b^* = q_2 \quad (20)$$

$$\left(b^*\right)\frac{\partial}{\partial_a} = b^*(b)\frac{\partial}{\partial_a} = \phi \tag{21}$$

$$\left(b^*\right)\frac{\partial}{\partial_b} = b^*(b)\frac{\partial}{\partial_b} = b^* \tag{22}$$

NFA obtained using the right partial derivative is shown in Fig. 5.2.

5.4 LINEAR EXPRESSION AND ITS APPLICATIONS IN BIO-MOLECULAR STRUCTURE

Sempere [75] introduced the concept of linear expression to represent linear language. Linear language is a sub-class of context-free languages. Furthermore, he introduced an approach for the conversion of a linear expression to linear grammar and vice versa. In this subsection, we will represent biological structures (Inverted repeats, Hairpin structure, Stem and Loop) by linear expression using Sempere approach [75].

Def. 5.4.1 [75]: Let $\Sigma = \{a_1, a_2, ..., a_n\}$ and $\Delta = \{l, r\}$ be alphabets, then Indexed alphabet is denoted by $\Sigma_\Delta = \{a_{1_l}, a_{1_r}, a_{2_l}, a_{2_r}, ..., a_{n_l}, a_{n_r}\}$.

Def. 5.4.2 [75]: Given Σ_Δ be indexed alphabet, then a linear expression can be defined using following rules:

- ϕ is a linear expression,
- ε is a linear expression,
- $\forall a \in \Sigma, a_l$ and a_r are linear expressions, and
- If r_1 and r_2 are linear expression, then $(r_1), r_1 + r_2, r_1 r_2, r_1^*$ are also linear expressions.

  Concatenation is performed using following rules:

  i) Given $w = a_l.w_1$ and $w \in \left(\Sigma_\Delta\right)^*$, then $image_{\Sigma_\Delta}(a_l.w_1) = a.image_{\Sigma_\Delta}(w_1)$.

  ii) Given $w = a_r.w_1$ and $w \in \left(\Sigma_\Delta\right)^*$, then $image_{\Sigma_\Delta}(a_r.w_1) = image_{\Sigma_\Delta}(w_1).a$.

  iii) Given $w = \varepsilon$, then $image_{\Sigma_\Delta}(\varepsilon) = \varepsilon$.

5.4.1 LINEAR EXPRESSION TO REPRESENT INVERTED REPEATS

In Inverted repeats, nucleotides sequence is followed by its reverse complement. Biomolecular structure for Inverted Repeats is shown in Fig. 5.3. For example, given initial sequence $5'taacg\,3'$ then its reverse complement is *cgtta* and inverted repeat

sequence becomes $5'taacg\ cgtta\ 3'$. It is a context-free grammar and its linear equivalent grammar is as follows:

$$S \to gA \mid tB \mid aX \mid cY \mid \varepsilon$$

$$A \to Sc$$

$$B \to Sa$$

$$X \to St$$

$$Y \to Sg$$

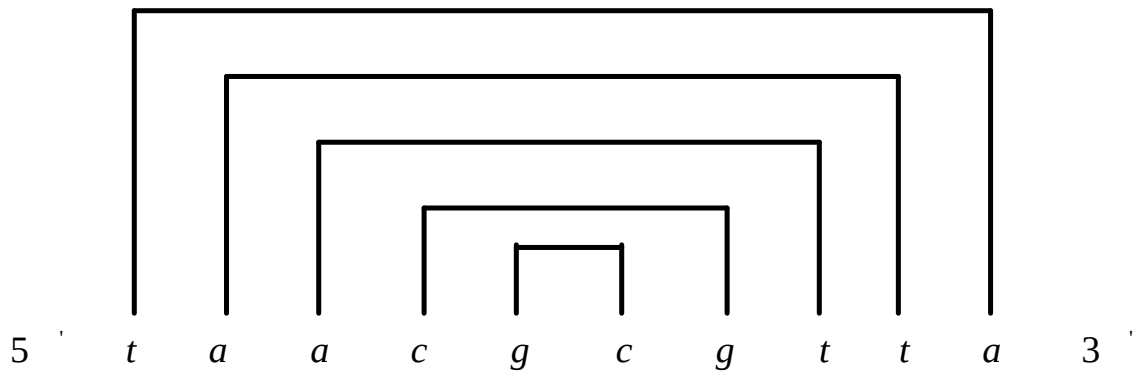

Fig. 5.3. Inverted Repeats structure

Grammar after rewriting in term of indexed alphabet:

$$S \to g_l A \mid t_l B \mid a_l X \mid c_l Y \mid \varepsilon$$

$$A \to c_r S$$

$$B \to a_r S$$

$$X \to t_r S$$

$$Y \to g_r S$$

Solving these, we get

$$S = g_l c_r S \mid t_l a_r S \mid a_l t_r S \mid c_l g_r S \mid \varepsilon$$

$$= (g_l c_r + t_l a_r + a_l t_r + c_l g_r) S \mid \varepsilon$$

$$= (g_l c_r + t_l a_r + a_l t_r + c_l g_r)^* \qquad (23)$$

5.4.2 LINEAR EXPRESSION FOR STEM AND LOOP

Bio-molecular structure for stem and loop is shown in Fig. 5.4. Here – represents the complementary pair.

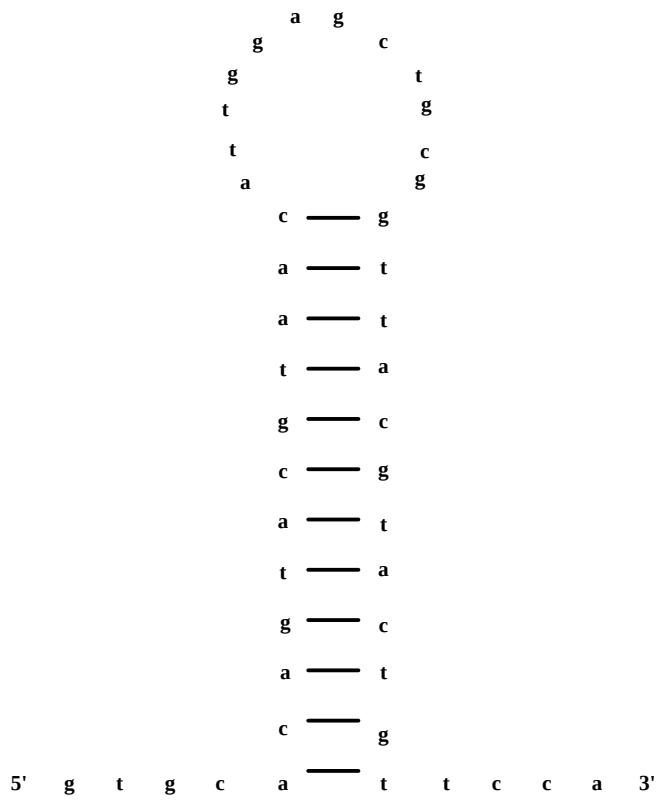

Fig. 5.4. Stem and Loop structure

Linear grammar for Stem and Loop:

$$S \to aS \mid gS \mid tS \mid cS \mid A$$

$$A \to Aa \mid Ag \mid At \mid Ac \mid B$$

$$B \to gB_1 \mid tB_2 \mid cB_3 \mid aB_4 \mid gB_5 \mid tB_5 \mid cB_5 \mid aB_5$$

$$B_1 \to Bc$$

$$B_2 \to Ba$$

$$B_3 \to Bg$$

$$B_4 \to Bt$$

$$B_5 \to gB_5 \mid tB_5 \mid cB_5 \mid aB_5 \mid \varepsilon$$

Solving these we get

$$S = a_l S + g_l S + t_l S + c_l S + A \tag{24}$$

$$A = a_r A + g_r A + t_r A + c_r A + B \tag{25}$$

$$B = g_l B_1 + t_l B_2 + c_l B_3 + a_l B_4 + g_l B_5 + t_l B_5 + c_l B_5 + a_l B_5 \tag{26}$$

$$B_1 \to c_r B \tag{27}$$

$$B_2 \rightarrow a_r B \tag{28}$$

$$B_3 \rightarrow g_r B \tag{29}$$

$$B_4 \rightarrow t_r B \tag{30}$$

$$B_5 = g_l B_5 + t_l B_5 + c_l B_5 + a_l B_5 + \varepsilon$$

$$= \left(g_l + t_l + c_l + a_l\right)^* \tag{31}$$

Putting value of B_1, B_2, B_3, B_4 and B_5 in B

$$B = g_l c_r B + t_l a_r B + c_l g_r B + a_l t_r B + \left(g_l + t_l + c_l + a_l\right) B_5$$

$$= \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right) B + \left(g_l + t_l + c_l + a_l\right)\left(g_l + t_l + c_l + a_l\right)^*$$

$$= \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right)^* \left(g_l + t_l + c_l + a_l\right)\left(g_l + t_l + c_l + a_l\right)^* \tag{32}$$

Putting value of B in A

$$A = \left(a_r + g_r + t_r + c_r\right) A + \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right)^* \left(g_l + t_l + c_l + a_l\right)\left(g_l + t_l + c_l + a_l\right)^*$$

$$= \left(a_r + g_r + t_r + c_r\right)^* \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right)^* \left(g_l + t_l + c_l + a_l\right)\left(g_l + t_l + c_l + a_l\right)^* \tag{33}$$

Putting value of A in S

$$S = a_l S + g_l S + t_l S + c_l S + A$$

$$= \left(a_l + g_l + t_l + c_l\right) S + A$$

$$= \left(a_l + g_l + t_l + c_l\right) S + \left(a_r + g_r + t_r + c_r\right)^* \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right)^* \left(g_l + t_l + c_l + a_l\right)$$

$$\left(g_l + t_l + c_l + a_l\right)^*$$

$$= \left(a_l + g_l + t_l + c_l\right)^* \left(a_r + g_r + t_r + c_r\right)^* \left(g_l c_r + t_l a_r + c_l g_r + a_l t_r\right)^* \left(g_l + t_l + c_l + a_l\right)$$

$$\left(g_l + t_l + c_l + a_l\right)^* \tag{34}$$

5.4.3 LINEAR EXPRESSION FOR HAIRPIN STRUCTURE

Bio-molecular hairpin structure is shown in Fig. 5.5. Context-free grammar for haipin structure is as follow:

$$S \rightarrow aSt \mid tSa \mid cSg \mid gSc \mid \varepsilon$$

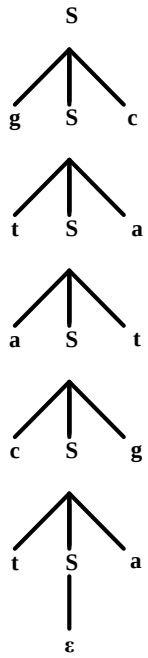

Fig. 5.5. Hairpin Structure

Its equivalent linear grammar is

$$S \to aA_1 \mid tA_2 \mid cA_3 \mid gA_4 \mid \varepsilon$$

$$A_1 \to St$$

$$A_2 \to Sa$$

$$A_3 \to Sg$$

$$A_4 \to Sc$$

Solving these, we get

$$S = a_l A_1 + t_l A_2 + c_l A_3 + g_l A_4 + \varepsilon \tag{35}$$

$$A_1 = t_r S \tag{36}$$

$$A_2 = a_r S \tag{37}$$

$$A_3 = g_r S \tag{38}$$

$$A_4 = c_r S \tag{39}$$

Putting value of A_1, A_2, A_3 and A_4 in S

$$S = a_l t_r S + t_l a_r S + c_l g_r S + g_l c_r S + \varepsilon$$

$$= \left(a_l t_r + t_l a_r + c_l g_r + g_l c_r\right) S + \varepsilon$$

$$= \left(a_l t_r + t_l a_r + c_l g_r + g_l c_r\right)^* \tag{40}$$

5.5 FUZZY LINEAR EXPRESSION AND FUZZY LINEAR GRAMMAR

In this subsection, the concept of fuzzy linear expressions and fuzzy linear grammars are explored. Conversion of linear grammars to linear expressions involve following two steps:

i. Convert a fuzzy linear grammar into fuzzy normal form.
ii. Find an equivalent linear expression to fuzzy normal form.

Def. 5.5.1: Given Σ_Δ be indexed alphabet, then a linear expression can be defined using following rules:

- ϕ is a linear expression,
- ε is a linear expression,
- $\forall a \in \Sigma$, a_l and a_r are linear expressions,
- If $a_l, a_r \in \Sigma_\Delta, \lambda \in (0,1]$, then $L(\lambda a_\ell) = \lambda a$ and $L(\lambda a_r) = \lambda a$, and
- If r_1 and r_2 are linear expressions, then $(r_1), r_1 + r_2, r_1 r_2, r_1^*$ are also linear expressions.

  Concatenation is performed using following rules:

  i. Given $w = \lambda_1 a_l . \lambda_2 w_1$ where $a_l w_1 \in (\Sigma_\Delta)^*$ and $\lambda_1, \lambda_2 \in (0,1]$ then $image_{\Sigma_\Delta}(\lambda_1 a_l . \lambda_2 w_1) = \lambda_1 a . image_{\Sigma_\Delta}(\lambda_2 w_1)$.

  ii. Given $w = \lambda_1 a_r . \lambda_2 w_1$ where $a_r w_1 \in (\Sigma_\Delta)^*$ and $\lambda_1, \lambda_2 \in (0,1]$ then $image_{\Sigma_\Delta}(\lambda_1 a_r . \lambda_2 w_1) = image_{\Sigma_\Delta}(\lambda_2 w_1) . \lambda_1 a .$

  iii. Given $w = \varepsilon$, then $image_{\Sigma_\Delta}(\varepsilon) = \varepsilon$.

Def. 5.5.2: A fuzzy linear grammar is in normal form if all the productions are of the form $A \xrightarrow{\lambda} aB \mid Ba \mid \varepsilon$ where $A, B \in N$, $a \in \Sigma$ and $\lambda \in (0,1]$.

A fuzzy linear grammar G can be converted into the normal form G' using the proposed algorithm 5.5.1.

Algorithm 5.5.1: Fuzzy_normal_form (G, G')

Input: Given grammar $G(N, T, S, P)$ is a fuzzy linear grammar such that $A \xrightarrow{\lambda} \alpha_1 B \alpha_2 \mid \alpha_3$ such that $\alpha_1, \alpha_2, \alpha_3 \in T^*$.

Output: Grammar $G^{'}(N^{'}, T, S, P^{'})$ in normal form with production rules of the form $A \xrightarrow{\lambda_1} aB \mid Ba \mid \varepsilon$ such that $L(G^{'}) = L(G)$.

$P^{'} \leftarrow \phi$

For each $p \in P$ **do**

If $(\alpha_1 \neq \varepsilon)$

Find the first terminal of α_1 say a.

$$P^{'} = P^{'} \bigcup \{A \xrightarrow{\lambda} aC \mid \alpha_1 B \alpha_2 = aC \wedge C \notin N^{'}\}$$

$$P = \{P - p\} \bigcup \{C \xrightarrow{1} \alpha_4 \mid \alpha_1 B \alpha_2 = a\alpha_4\}$$

Else

If $(\alpha_2 \neq \varepsilon)$

Find the last terminal of α_2 say a.

$$P^{'} = P^{'} \bigcup \{A \xrightarrow{\lambda} Ca \mid B\alpha_2 = Ca \wedge C \notin N^{'}\}$$

$$P = \{P - p\} \bigcup \{C \xrightarrow{1} \alpha_4 \mid B\alpha_2 = \alpha_4 a\}$$

Else

//No action will be taken

End If

End If

End of loop

The proposed algorithm is applied on a linear grammar in Example 5.3.

Example 5.3: Consider the fuzzy linear grammar (N, T, S, P) where $T = \{0, 1\}$, $N = \{S\}$, and P is given as follows:

$$S \xrightarrow{0.5} 0S0$$

$$S \xrightarrow{0.4} 1S1$$

$$S \xrightarrow{1} \varepsilon$$

Consider $S \xrightarrow{0.5} 0S0$

$P^{'} \leftarrow \phi$

Now $P' = P' \bigcup \{A \overset{\lambda}{\rightarrow} aC \mid \alpha_1 B \alpha_2 = aC \wedge C \notin N'\}$

$P' = P' \bigcup \{S \overset{0.5}{\rightarrow} 0A\}$ $\qquad\qquad$ $P = \{P - p\} \bigcup \{A \overset{1}{\rightarrow} S0\}$

Similarly $S \overset{0.4}{\rightarrow} 1S1$

$P' = P' \bigcup \{S \overset{0.4}{\rightarrow} 1B\}$ $\qquad\qquad$ $P = P \bigcup \{B \overset{1}{\rightarrow} S1\}$

Considering $S \overset{1}{\rightarrow} \varepsilon$, $A \overset{1}{\rightarrow} S0$ and $B \overset{1}{\rightarrow} S1$ all these are already in normal form. Hence added to P'.

Now, we can convert a fuzzy normal form to fuzzy linear expression by extending the concept for the conversion of linear grammar to linear expression.

Example 5.4: Consider the fuzzy context-free language $L = \{a^n b^m \mid n, m \geq 0\}$ with the following properties

$$L = \begin{Bmatrix} a^n b^m \mid n = m \wedge F(w) = 1 \\ a^n b^m \mid n > m \wedge F(w) = 0.7 \\ a^n b^m \mid n < m \wedge F(w) = 0.5 \end{Bmatrix}$$

where $F(w)$ denotes the membership value of the string w.

Fuzzy linear grammar for the language L:

$S \overset{1}{\rightarrow} aA$

$S \overset{1}{\rightarrow} \varepsilon$

$A \overset{1}{\rightarrow} Sb$

$A \overset{0.7}{\rightarrow} aA$

$A \overset{0.5}{\rightarrow} Ab$

Consider $w = aabb$

These production rules can be written as

$$S = a_l A + \varepsilon \tag{41}$$

$$\begin{aligned} A &= b_r S + 0.7 a_l A + 0.5 b_r A \\ &= b_r S + (0.7 a_l + 0.5 b_r) A \\ &= (0.7 a_l + 0.5 b_r)^* b_r S \end{aligned} \tag{42}$$

Putting values of (42) in (41), we get

$$S = a_l A + \varepsilon$$

$$S = (a_l (0.7a_l + 0.5b_r)^* b_r)S + \varepsilon$$

$$S = (a_l (0.7a_l + 0.5b_r)^* b_r)^* \quad (43)$$

Membership of a string can be determined using max-min composition. Consider $w = aabb$.

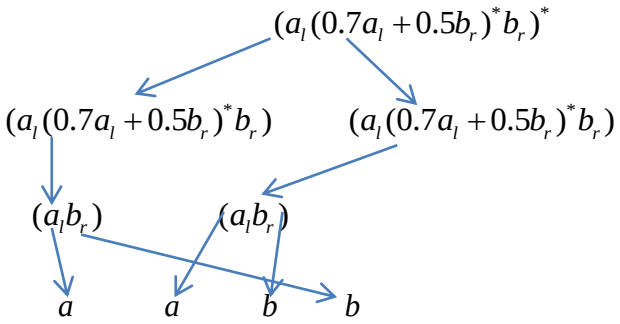

Fig. 5.6. Generation of a string $w = aabb$ from fuzzy linear expression

Example 5.5: Consider the context-free language $L = \{ww^r \mid w \in (a,b)^*\}$ with the following properties:

$$S \xrightarrow{1} aA \mid bB \mid \varepsilon$$

$$A \xrightarrow{1} Sa$$

$$B \xrightarrow{1} Sb$$

Error productions

$$A \xrightarrow{0.8} aS$$

$$A \xrightarrow{0.6} Sb$$

$$B \xrightarrow{0.7} bS$$

$$B \xrightarrow{0.5} Sa$$

These productions can be converted into

$$S = a_l A + b_l B + \varepsilon \quad (44)$$

$$A = a_r S + 0.8a_l S + 0.6b_r S \quad (45)$$

$$B = b_r S + 0.7b_l S + 0.5a_r S \quad (46)$$

Putting values of (45)and (46) in (44), we get

$$S = a_l(a_r S + 0.8a_l S + 0.6b_r S) + b_l(b_r S + 0.7b_l S + 0.5a_r S) + \varepsilon$$

$$S = (a_l a_r + a_l 0.8a_l + a_l 0.6b_r + b_l b_r + b_l 0.7b_l + b_l 0.5a_r)S + \varepsilon$$

$$S = (a_l a_r + a_l 0.8a_l + a_l 0.6b_r + b_l b_r + b_l 0.7b_l + b_l 0.5a_r)^* \quad (47)$$

5.6 PROPOSED METHODOLOGY

In this section, the extended partial derivatives have been applied to the fuzzy regular expressions. We consider that the membership value $\lambda \in [0,1]$ and Σ are disjoint. To keep it simple, we consider $\Sigma = \{a, b\}$.

$\partial_a(\lambda\varepsilon) = \phi$, (48)

$\partial_a(\lambda\phi) = \phi$, (49)

$\partial_a(\lambda a) = \lambda\partial_a(a) = \lambda\varepsilon$, (50)

$\partial_a(\lambda b) = \phi$, (51)

$\partial_a(\lambda_1 r_1 \bigcup \lambda_2 r_2) = \lambda_1\partial_a(r_1) \bigcup \lambda_2\partial_a(r_2)$, (52)

$\partial_a(\lambda r_1^*) = \lambda\partial_a(r_1)r_1^*$, (53)

$\partial_a(\lambda r_1)^* = \lambda\partial_a(r_1)(\lambda r_1)^*$, (54)

If $\varepsilon \in L(r_1)$

then $\partial_a(\lambda_1 r_1 \lambda_2 r_2) = \lambda_1\partial_a(r_1)\lambda_2 r_2 \bigcup \lambda_1\partial_a(\lambda_2 r_2)$

else $\partial_a(\lambda_1 r_1 \lambda_2 r_2) = \lambda_1\partial_a(r_1)\lambda_2 r_2$, (55)

Starting state is defined by $q_0 = r$,

and its membership value is represented by $\lambda = 1$, (56)

Final state of the *NFA* is defined by $F = \{q \mid q \in Q \wedge q = \lambda r \wedge \varepsilon \in LR(r)\}$. (57)

5.7 NUMERICAL EXAMPLES

In this section, the proposed methodology is applied on two numerical examples.

Example 5.6: Consider the fuzzy regular expression $r = (0.2a^*)(ba + 0.9b)^*$

$\partial_a(r) = \partial_a((0.2a^*)(ba + 0.9b)^*) = \partial_a((0.2a^*)(ba + 0.9b)^*)$

$= 0.2\partial_a((a^*)(ba + 0.9b)^*) \bigcup 0.2(\partial_a((ba + 0.9b)^*))$

$= 0.2\partial_a(a)(a^*)(ba + 0.9b)^* \bigcup 0.2\partial_a(ba + 0.9b)(ba + 0.9b)^*$

$= 0.2(a^*)(ba + 0.9b)^* \bigcup 0.2\partial_a(ba)(ba + 0.9b)^* \bigcup 0.2\partial_a(0.9b)(ba + 0.9b)^*$

$= 0.2(a^*)(ba+0.9b)^* \cup \phi \cup \phi$

$\partial_b(r) = \partial_b((0.2a^*)(ba+0.9b)^*) = \partial_b((0.2a^*)(ba+0.9b)^*)$

$= 0.2\partial_b((a^*)(ba+0.9b)^*) \cup 0.2\partial_b((ba+0.9b)^*)$

$= 0.2\partial_b(a)(a^*)(ba+0.9b)^* \cup 0.2\partial_b(ba+0.9b)(ba+0.9b)^*$

$= 0.2\phi(ba+0.9b)^* \cup 0.2\partial_b(ba)(ba+0.9b)^* \cup 0.2\partial_b(0.9b)(ba+0.9b)^*$

$= \phi \cup 0.2a(ba+0.9b)^* \cup 0.18\partial_b(b)(ba+0.9b)^*$

$= \phi \cup 0.2a(ba+0.9b)^* \cup 0.18(ba+0.9b)^*$

Now consider $r_1 = (a^*)(ba+0.9b)^*$

$\partial_a(r_1) = \partial_a(a^*)(ba+0.9b)^*$

$= \partial_a(a^*)(ba+0.9b)^* \cup \partial_a(ba+0.9b)^*$

$= \partial_a(a)(a^*)(ba+0.9b)^* \cup \partial_a(ba+0.9b)^*$

$= (a^*)(ba+0.9b)^* \cup \phi$

$\partial_b(r_1) = \partial_b(a^*)(ba+0.9b)^*$

$= \partial_b(a^*)(ba+0.9b)^* \cup \partial_b(ba+0.9b)^*$

$= \phi \cup \partial_b(ba+0.9b)(ba+0.9b)^*$

$= \partial_b(ba)(ba+0.9b)^* \cup \partial_b(0.9b)(ba+0.9b)^*$

$= (a)(ba+0.9b)^* \cup 0.9(ba+0.9b)^*$

For $r_2 = a(ba+0.9b)^*$

$\partial_a(r_2) = \partial_a(a(ba+0.9b)^*)$

$= (ba+0.9b)^*$ $\partial_b(r_2) = \phi$

For $r_3 = (ba+0.9b)^*$

$\partial_a(r_3) = \partial_a((ba+0.9b)^*)$

$= \partial_a(ba+0.9b)^*$

$= \partial_a(ba+0.9b)(ba+0.9b)^* = \phi$

$\partial_b(r_3) = \partial_b((ba+0.9b)^*)$

$= \partial_b((ba+0.9b)^*)$

$= \partial_b (ba + 0.9b)(ba + 0.9b)^*$

$= \partial_b (ba)(ba + 0.9b)^* \bigcup \partial_b (0.9b)(ba + 0.9b)^*$

$= (a)(ba + 0.9b)^* \bigcup 0.9(ba + 0.9b)^*$

Final state $F = \{r / 0.2, r_1 / 1, r_3 / 1\}$. Fig. 5.7 depicts the ε -free fuzzy automata corresponding to $r = (0.2a^*)(ba + 0.9b)^*$.

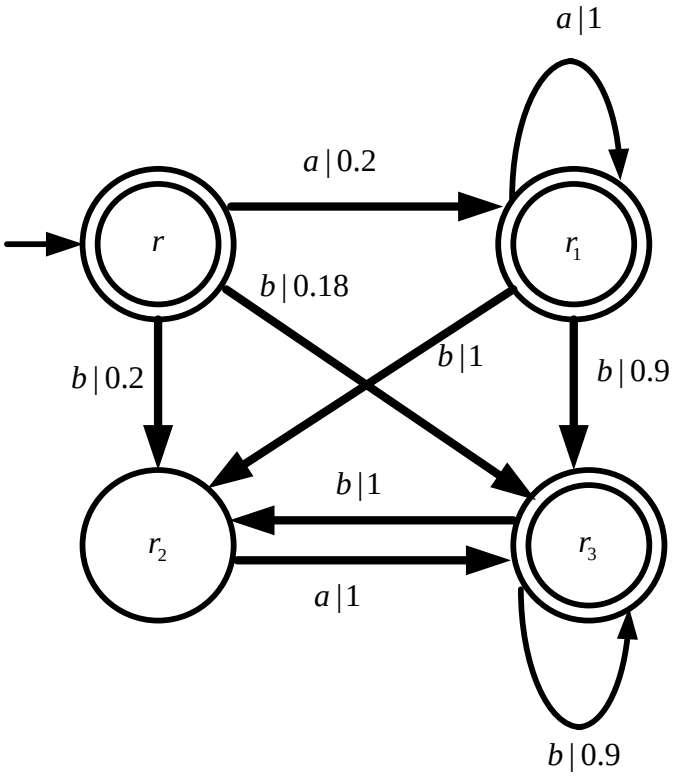

Fig. 5.7: ε -free fuzzy automata for $r = (0.2a^*)(ba + 0.9b)^*$

Example 5.7: Consider the fuzzy regular expression $r = 0.2((0.1(xy)^*)^* + y)$ [17-18]. It follows that

$r = 0.02(xy)^* + 0.2\varepsilon + 0.2y = 0.02(xy)^* + 0.2\varepsilon + 0.2y$

$\partial_x (r) = \partial_x (0.02(xy)^* + 0.2\varepsilon + 0.2y)$

$= 0.02y(xy)^* + \phi + \phi$

$\partial_y (r) = \partial_y (0.02(xy)^* + 0.2\varepsilon + 0.2y)$

$= \phi + \phi + 0.2\varepsilon$

$\partial_y (r_1) = \partial_y (y(xy)^*) = (xy)^*$

$r_2 = \varepsilon$

$\partial_x(r_2) = \partial_y(r_2) = \phi$

$r_3 = (xy)^*$

$\partial_x((xy)^*) = y(xy)^* = r_1$

$\partial_y(r_2) = \phi$

Final state $F = \{r/0.2, r_1/1, r_3/1\}$. Fig. 5.8 depicts the ε -free fuzzy automata corresponding to $r = 0.2((0.1(xy)^*)^* + y)$.

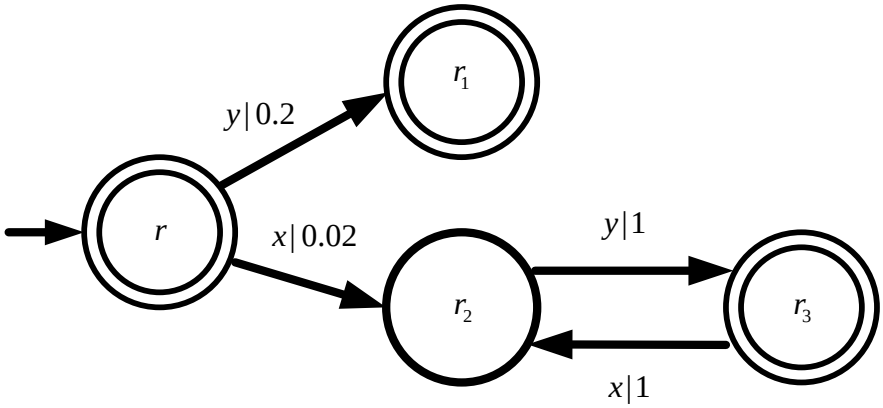

Fig. 5.8: ε -free fuzzy automata for $r = 0.2((0.1(xy)^*)^* + y)$

5.8 APPLICATION OF TWO SIDED DERIVATIVES TO FUZZY LINEAR EXPRESSION

In this section, we will apply the concept of two-sided derivatives to the fuzzy linear expression. We consider two R/W head, one moving left to right and another right to left. Following rules are used to apply two-sided derivative:

i. Left derivative: Indexed symbol marked with l is used to read the string from left to right.

ii. Right derivative: Similarly, Indexed symbol marked with r is used to read the string from right to left.

Example 5.8. Consider the fuzzy linear expression of Example 5.5.

$r = (a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$

Consider $q_0 = (a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$

$\partial_{a_l, a_r}(r) = \partial_{a_l, a_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$

$$= \partial_{a_l, a_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)\, r$$

$$= \partial_{a_l, a_r}(a_l a_r)\, r = \partial_{a_r}(a_r)\, r = r = q_0$$

$$\partial_{a_l, b_r}(r) = \partial_{a_l, b_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$$

$$= \partial_{a_l, b_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)\, r$$

$$= \partial_{a_l, b_r}(a_l 0.6 b_r)\, r = r = q_0$$

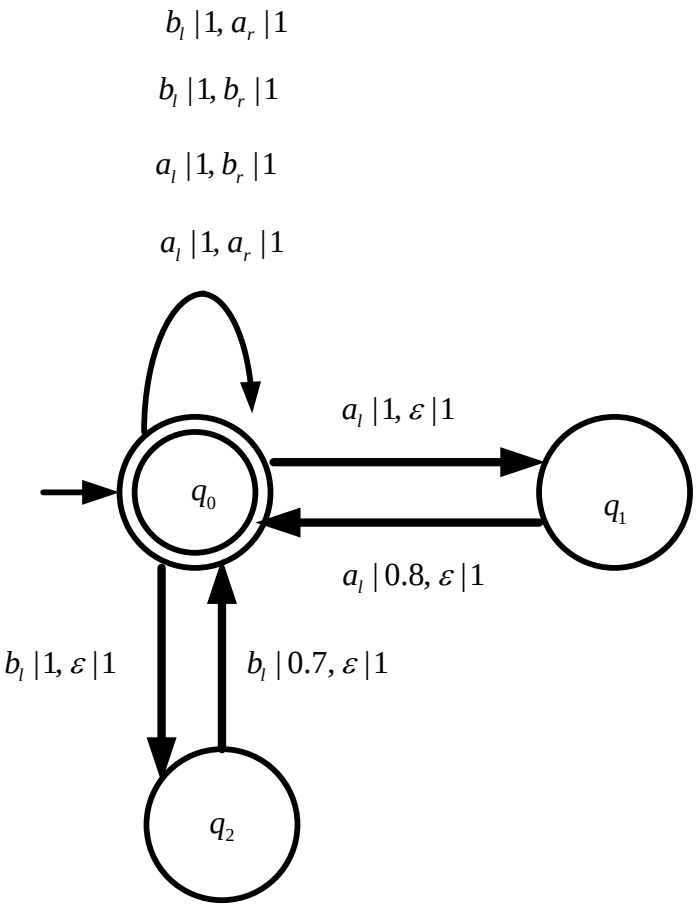

Fig. 5.9: Finite Automata using left and right side derivative

$$\partial_{b_l, b_r}(r) = \partial_{b_l, b_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$$

$$= \partial_{b_l, b_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)\, r$$

$$= \partial_{b_l, b_r}(b_l b_r)\, r = r = q_0$$

$$\partial_{b_l, a_r}(r) = \partial_{b_l, a_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)^*$$

$$= \partial_{b_l, a_r}(a_l a_r + a_l 0.8 a_l + a_l 0.6 b_r + b_l b_r + b_l 0.7 b_l + b_l 0.5 a_r)\, r$$

$$= \partial_{b_l, a_r}(b_l 0.5 a_r)\, r = r = q_0$$

$$\partial_{a_l,\varepsilon}(r) = \partial_{a_l,\varepsilon}(a_l a_r + a_l 0.8a_l + a_l 0.6b_r + b_l b_r + b_l 0.7b_l + b_l 0.5a_r)^*$$

$$= \partial_{a_l,\varepsilon}(a_l a_r + a_l 0.8a_l + a_l 0.6b_r + b_l b_r + b_l 0.7b_l + b_l 0.5a_r)\, r$$

$$= \partial_{a_l,\varepsilon}(a_l 0.8a_l)\, r = 0.8a_l\, r = q_1$$

$$\partial_{b_l,\varepsilon}(r) = \partial_{b_l,\varepsilon}(a_l a_r + a_l 0.8a_l + a_l 0.6b_r + b_l b_r + b_l 0.7b_l + b_l 0.5a_r)^*$$

$$= \partial_{b_l,\varepsilon}(b_l 0.7b_l)\, r = 0.7b_l\, r = q_2$$

$$\partial_{a_l,\varepsilon}(r) = \partial_{a_l,\varepsilon}(0.8a_l\, r) = \partial_{a_l,\varepsilon}(0.8a_l)\, r = r = q_0$$

$$\partial_{b_l,\varepsilon}(r) = \partial_{b_l,\varepsilon}(0.7b_l\, r) = \partial_{b_l,\varepsilon}(0.7b_l)\, r = r = q_0$$

Fig. 5.9 represents the fuzzy automata obtained using left and right derivatives.

5.9 RESULTS AND DISCUSSIONS

Theorem 5.1: Let r be a fuzzy regular expression, then the ε-free fuzzy automata M is obtained by using the extended partial derivative approach such that $LR(M) = LR(r)$.

Proof. We prove by induction on the number of instances of symbols in r. We consider the membership value of the starting state q_0 as 1.

Basis Step: The following two cases can occur:

Case 1: For $m = 0$. The following two sub- cases are possible.

Case 1.1: If $r = \lambda\phi$, then $LR(r) = \phi$. Then using extended partial derivatives, $\forall a \in \Sigma,\ \partial_a(\lambda r) = \lambda\partial_a(r) = \lambda\phi = \phi.$ The $NFA\ M = \{\{q_0, q_f\}, \Sigma, \delta, q_0/1, q_f/1\}$ is represented in Fig. 5.9. Thus $LR(M) = \phi.$

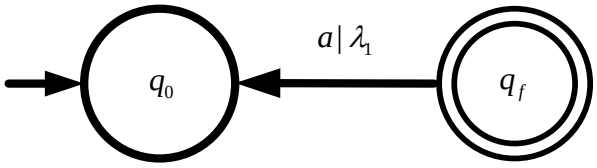

Fig. 5.9: *NFA* for $r = \lambda\phi$

Case 1.2: If $r = \lambda\varepsilon$, then $LR(r) = \{\varepsilon/\lambda\}$, so $\partial_a(\lambda\varepsilon) = \lambda\partial_a(r) = \lambda\phi = \phi, \forall a \in \Sigma$. The $NFA\, M = \{\{q_0\}, \Sigma, \delta, q_0/1, q_0/\lambda\}$ is represented in Fig. 5.10.

Thus $LR(M) = \{\varepsilon/\lambda\}$.

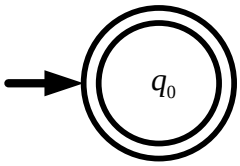

Fig. 5.10: *NFA* for $r = \lambda\varepsilon$

Case 2: If $m = 1$ and $r = \lambda a$ where $\lambda \in L, a \in \Sigma$, then $LR(r) = \{a / \lambda\}$. Using $\partial_a(\lambda a) = \lambda\partial_a(a) = \lambda\varepsilon$ and for $\forall b \in \Sigma \wedge b \neq a,$ we have $\partial_b(\lambda a) = \lambda\partial_b(a) = \phi.$ The *NFA* $M = \{\{q_0, q_1\}, \Sigma, \delta, q_0 / 1, q_1 / 1\}$ is represented in Fig. 5.11. Thus $LR(M) = \sigma \circ \delta_a \circ \tau = \{\lambda a\}.$ This completes the basis step.

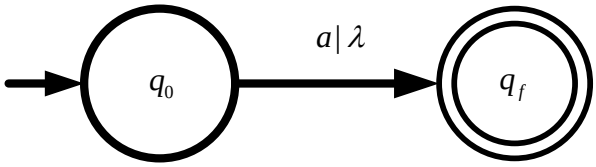

Fig. 5.11: *NFA* for $r = \lambda a$

Induction hypothesis: Let $r \in LR$ such that the total number of instances of the symbol in r is m and let *NFA* M be designed using the proposed approach such that $LR(r) = LR(M)$.

Induction step: Let fuzzy regular expression r_1 such that the total number of instances of symbols in r_1 is $m+1$, and it is obtained by adding any arbitrary $a \in \Sigma$ with membership value λ_{m+1} to r .

Following cases can occur:

Case 1: Consider $r_1 = r\lambda_{m+1}a$, on applying extended partial derivatives to r_1 . Similar states are obtained by applying extended partial derivatives to r except that the fuzzy regular expression of each state is concatenated by $\lambda_{m+1}a$. Using the extended rule for concatenation *NFA* M can be converted into *NFA* M' by adding a new state q_f^1 such that $\exists q \in q_f$ there will be an outgoing edge from q to q_f^1 labeled with a / λ_{m+1} .

Case 2: Consider $r_1 = r \bigcup \lambda_{m+1}a$, on applying extended partial derivatives to r_1 . Using the extended rule for the union, a new state (say q_f^1) representing the fuzzy regular expression

ε (if already not existing) will be added. *NFA M* can be converted into *NFA M'* by adding an edge from q_0 (starting state) to new state q_f^1 labeled with a/λ_{m+1}. Other cases using union operation is trivially true.

Case 3: Consider $r_1 = r\lambda_{m+1}a^*$. By applying extended partial derivatives to r, each final state q is representing a fuzzy regular expression and $\varepsilon \in q$. Following two sub-cases can occur:

Case 3.1: In this sub-case, we consider self-loop at final state q. Using the extended concatenation and Kleene closure rules a new state q_f^1 will be added to Q, such that a new edge from q to q_f^1 labeled with a/λ_{m+1}. Furthermore, a self-loop on q_f^1 labeled with a/λ_{m+1} also occurs and also the new final state q_f^1 will be added to the set of the final states.

Case 3.2: In this sub-case, we consider no self-loop at final state q. A self-loop with the transition a/λ_{m+1} will be added and q remains the final state using the proposed rules.

Based on the above observations, we observe that the induction step follows in the extended partial derivatives. This completes the proof of the theorem. □

Lemma 5.1. For any fuzzy regular expression r, the size of fuzzy automaton obtained using the proposed approach is smaller than the size of ε-fuzzy automata obtained from r using the Stamenkovic and Ćirić approach.

Proof. Stamenkovic and Ćirić approach [17] is based on the concept of position automata, whereas the proposed approach is based on partial derivatives. Champarnaudand Ziadi [53] proved that partial derivative automaton is a quotient of the partial derivative automaton. Hence the ε-free automaton obtained using the proposed approach is a quotient of the automaton obtained by Stamenkovic and Ćirić approach [17] using position automata. Although, Stamenkovic and Ćirić reduced the obtained automata. Further, they applied the reduction procedure to reduce the number of states. This completes the proof of the lemma. □

Table 5.1 illustrates the difference between the proposed approach and the existing approaches in the literature. From Table 5.1, it is evident that the proposed approach will yield less number of states as compared with the existing approaches in the literature.

Given a fuzzy regular expression r having length n and m denotes the number of occurrences of symbols from Σ in r. Stamenkovic and Ćirić [17] first converted the fuzzy regular expressions into fuzzy automata with $\varepsilon-$ fuzzy automata using the position automata. Further, they have converted it into $\varepsilon-$ free fuzzy automata.

Table 5.1. Comparison between existing and proposed approach

Attribute	**Stamenkovic and Ćirić Approach [17]**	**Proposed Approach**
Approach based on	Position Automata	Antimirov's partial derivatives
Number of states	Exactly $m+1$ states and further reduction procedure is required.	$\leq m+1$ states
Output	ε-fuzzy Automata	ε-free fuzzy Automata

In comparison, our approach converts the fuzzy regular expression into ε-free fuzzy automata directly using the partial derivatives approach.

5.10 CONCLUSION

In this chapter, we have applied the concept of linear expression to represent Bio-molecular structure. We have introduced the concept of fuzzy linear expression. Further, an algorithm is designed to convert a fuzzy linear grammar to fuzzy linear expression. Further, we have extended Antimirov's partial derivatives for the conversion of fuzzy regular expressions to fuzzy automata. The number of states of the ε-free fuzzy automata is less than the number of states resulting from Stamenkovic and Ćirić approach [17], as shown in Table 5.1. The concept of two-sided derivatives has been applied to fuzzy linear expressions. Finally, the accuracy of the conversion is demonstrated by a numerical example.

Chapter-6

CONCLUSION AND FUTURE SCOPE

6.1 SUMMARY OF THE MAIN CONTRIBUTION

In this dissertation, we proposed some improvements in the construction of fuzzy automata. The summary of the main contribution is presented as follows:

1. Survey of existing literature on regular expression, finite automata, fuzzy regular expression, fuzzy automata, and their applications. We have explored the existing literature on conversion of regular expression to finite automata and vice-versa. Existing literature on the conversions of fuzzy regular expression to fuzzy finite automata and vice-versa is also explored. Further, we have reviewed the existing approaches for the conversion of parallel regular expression to a finite automaton. In addition to this, we also explored the various application areas of fuzzy languages and automata.
2. Introduced the concept of parallel fuzzy regular expression.
3. Designed an algorithm for the conversion of parallel fuzzy regular expression to epsilon-free non-deterministic fuzzy automaton. In this work, we have used asynchronous interleaving shuffle operator.
4. Extended transitive closure for the conversion of fuzzy automaton into a fuzzy regular expression.
5. Introduced the concept of fuzzy linear expression.
6. Concept of linear expression is applied to represent Bio-molecular structure.
7. Revised Antimirov's partial derivatives for the conversion of fuzzy regular expression to $\varepsilon-$ free fuzzy non-deterministic finite automaton.
8. Concept of two-sided derivatives has been applied to a fuzzy linear expression.

6.2 FUTURE SCOPE

Following are a number of promising and stimulating directions for future research building on the proposed work and gleaning of knowledge presented in this dissertation.

7.2.1 INTUITIONISTIC FUZZY AUTOMATA AND FUZZY REGULAR EXPRESSIONS

Significant research has been done on intuitionistic fuzzy automata and intuitionistic fuzzy regular expression, but no research work has been done on their conversions counterpart. Algorithms can be designed for the conversion of intuitionistic fuzzy regular expression to intuitionistic fuzzy automata and vice-versa.

7.2.2 CONCEPT OF FUZZY LINEAR EXPRESSION TO REPRESENT FUZZY CONTEXT-SENSITIVE LANGUAGES

We have introduced the concept of fuzzy linear expressions to represent some fuzzy context-free languages. There is a scope to introduce different concept in indexed alphabet to represent fuzzy context-sensitive languages.

7.2.3 GENERAL SHUFFLE OPERATION ON FUZZY REGULAR LANGUAGES

There exists a variety of shuffle operators such as plain shuffling, weak synchronized shuffling, strong synchronized shuffling, synchronized composition, iterated shuffling. We have applied the concept of asynchronous interleaving to fuzzy regular expression. The proposed concept can be extended to general synchronous shuffling.

7.2.4 STATE COMPLEXITY ANALYSIS OF FUZZY COUPLED AUTOMATA

We have applied the concept of two sided derivatives to fuzzy linear expression. Automaton counterpart of linear expression is known as coupled automata. There is a scope to carry out state complexity analysis of the fuzzy coupled automata with respect to the length of input string w.

7.2.5 FUZZINESS IN REGULATED AUTOMATA

A number of regulated automata have been proposed such as Jumping Automata, Deep Pushdown Automata, Absolutely Unlimited Deep Pushdown Automata and Parallel Deep Pushdown Automata. Work can be carried out by the inclusion of fuzziness in these regulated automata.

7.2.6 CONVERSION OF FUZZY AUTOMATA FROM FUZZY REGULAR EXPRESSIONS

In this dissertation, we have applied the concept of Transitive closure for the conversion of fuzzy automata to fuzzy regular expression. Various other techniques such as state

elimination method, Arden theorem concept can be used for obtaining fuzzy regular expression.

7.2.7 TOOL DESIGN FOR LINEAR EXPRESSION

We have introduced the concept of linear expression to represent the Bio-molecular structure. Software can be developed for finding these structures in biological sequences.

REFERENCES

1. L. A. Zadeh, Fuzzy Sets, Information and Control, 8(3), 338-353, 1965.
2. E. S. Santos, Maximin automata, Information and Control, 13(4), 363-377, 1968.
3. A. V. Aho, R. Sethi and J. D. Ullman, Compilers: Principles, techniques and tools, Pearson Education, 2005.
4. A virus scanning system, Clam Antivirus: open source anti-virus toolkit, http://www.clamav.net/lang/en.
5. Intrusion detection systems, Firekeeper: Detect and block malicious sites http://firekeeper.mozdev.org
6. M. Nivat, Behaviors of synchronized system of processes, L. I. T. P. Report no 81-64, University de Paris 7, 1981.
7. P. D. Stotts and W. Pugh, Parallel finite automata for modeling concurrent software systems, Journal of Systems and Software, 27(1), 27-43, 1994.
8. W. G. Wee and K. S. Fu, A formulation of fuzzy automata and its application as a model of learning systems, IEEE Transaction on Systems Science and Cybernetics, 5(3), 215-223, 1969.
9. Thakur and D. Mishra, Fuzzy contrast mapping for image enhancement, 2nd International Conference on Signal Processing and Integrated Networks (SPIN), IEEE Conference Noida, 2015.
10. M. Ying, A formal model of computing with words, IEEE Transaction on Fuzzy System, 10(5), 640-652, 2002.
11. D. Qiu and H. Wang, A probabilistic model of computing with words, Journal of Computer and System Sciences, 70, 176-200, 2005.
12. D. Qiu, Supervisory control of fuzzy discrete event systems: a formal approach, IEEE Transactions on Systems, Man, and Cybernetics, Part B(Cybernetics), 35(1), 72-88, 2005.
13. L. A. Zadeh, Fuzzy Languages and their Relation to Human and Machine Intelligence. Proc. International Conference on Man and Computer, Bordeaux, France, 130-165, 1971.
14. J. N. Mordeson and D. S. Malik, Fuzzy Automata and Languages: Theory and Applications, Chapman and Hall/ CRC, Boca Raton, London, Newyork, Washington DC, 2002.

15. F. Lin and H. Ying, Modeling and control of fuzzy discrete event systems, IEEE Transactions on Systems, Man, and Cybernetics, Part B(Cybernetics), 32(4), 408-415, 2002.
16. N. Martinel, C. Micheloni and G. L. Foresti, A Pool of Multiple Person Re-Identification Experts, Pattern Recognition Letters, 71, 23-30, 2016.
17. A. Stamenković and M.Ćirić, Construction of fuzzy automata from fuzzy regular expressions, Fuzzy Sets and Systems, 199, 1-27, 2012.
18. J. R. G. D. Mendivil and J. R. Garitagoitia, A comment on "Construction of fuzzy automata from fuzzy regular expressions", Fuzzy Sets and Systems, 262, 102-110, 2015.
19. R. Kumar and A. Kumar, Metamorphosis of fuzzy regular expressions to fuzzy automata using the follow automata, preprint arXiv:1411.2865, 2014.
20. Y. Li, Finite automata theory with membership values in lattices, Information. Sciences, 181(5), 1003-1017, 2011.
21. Y. Li and W. Pedrycz, Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids, Fuzzy Sets andSystems, 156, 68-92, 2005.
22. Z. Li, P. Li and Y. M. Li, The relationships among several types of fuzzy automata, Information Sciences, 176(15), 2208-2226, 2006.
23. M. Ćirić, A. Stamenkovic, J. Ignjatovic and T. Petkovic, Fuzzy relation equations and reduction of fuzzy automata, Journal of Computer and System Sciences, 76(7), 609-633, 2010.
24. J. Ignjatovic, M. Ćirić and S. Bogdanovic, Determinization of fuzzy automata with membership values in complete residuated lattices, Information Sciences, 178(1), 164-180, 2008.
25. G. Grätzer, General lattice theory, Second edition, Springer Science and Business Media, 2002.
26. S. Kakoty, M. Lal and S. K. Sarma, Fuzzy expert logic to evaluate learner's expertise level in e-learning environment, IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Syed Ammal Engineering College, Ramanathapuram, India,23-25 August 148-152, 2012.
27. W. Okamoto, S. Tano, A. Inoue and R. Fujioka, A rule-based inference method for natural language propositions involving fuzzy quantifiers and truth qualifiers,

Journal of Japanese Society for Fuzzy Theory and Intelligent Informatics (SOFT), 19 (4), 412-420, 2007.

28. W. Okamato, S. Tano, T. Iwatani and A. Inoue, An effect by qualification of truth-qualifiers on inference results for natural language propositions involving fuzzy quantifiers, Journal of Japanese Society for Fuzzy Theory and Systems, 9 (3), 419-425, 1997.
29. V. M. Glushkov, The abstract theory of automata, Russian Mathematical Surveys, 16(5), 1961.
30. R. McNaughton and H. Yamada, Regular expressions and state graphs for automata, IRE Transactions on Electronic Computers, 9(1), 39-47, 1960.
31. D. Kuske, Schützenberger's theorem on formal power series follows from Kleene's theorems, Theoretical Computer Science, 401, 243-248, 2008.
32. B. D. Estrade, A. L. Perkins and J. M. Harris, Explicitly Parallel Regular Expressions, First International Multi-symposiums on Computer and Computational Sciences (IMSCCS06), IEEE, Hangzhou, China, 402-409, 2006.
33. B. D. Estrade, An investigation of equivalent serialized forms of parallel finite automata, Master's Thesis, The University of Southern Mississippi, USA, 2005.
34. J. C. Baeten and W. P. Weijland, Process Algebra, Cambridge tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, 18(1), 1990.
35. J. Clark, and M. Murata, RELAX NG specifications. http://relaxng.org/spec-20011203.html (accessed December 3, 2001), 2001.
36. K. Iwama, Unique Decomposability of Shuffled Strings: A Formal Treatment of Asynchronous Time-multiplexed Communication. Proc. 15th Annual ACM Symp. Theory of Computing, Boston, MA, USA, April 25-27, 374-381, 1983.
37. A. Kumar and A.K. Verma, A novel algorithm for the conversion of shuffle regular expressions into non-deterministic finite automata, Maejo International Journal Science and Technology, 7, 396-407, 2013.
38. A. Kumar and A.K. Verma, A novel algorithm for the conversion of parallel regular expressions to non-deterministic finite automata, Applied Mathematics and Information Science, 8, 95-105, 2014.
39. W. Gelade, Succinctness of regular expressions with interleaving, intersection and counting, Theoretical Computer Science, 411, 2987-2998, 2010.

40. J. E. Hopcroft, R. Motwani, and J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Pearson Education, Singapore, 2000.
41. S. C. Kleene, Representation of events in nerve nets and finite automata, In Automata Studies, pages 3-40. Ann. of Math. Studies No. 34, Princeton University Press, Princeton, NJ, 1956.
42. J. A. Brzozowski, Derivatives of regular expressions, Journal of the ACM, 11(4): 481-494, 1964.
43. D. Du and K. Ko, Problem Solving in Automata, Languages, and Complexity. John Wiley & Sons, New York, NY, 2001.
44. C. Neumann, Converting deterministic finite automata to regular expressions. Available on URL: http://neumannhaus. com/christoph/papers/2005-03-16. DFA to RegEx. Pdf, 2005.
45. H. Yamamoto, A New Recognition Algorithm for Extended Regular Expressions, International Symposium on Algorithms and Computation, ISAAC 2001 Christchurch, New Zealand, December 19-21, Lecture Notes in Computer Science 2223, 257-267, Springer, Berlin, Heidelberg, 2001.
46. H. Yamamoto, A New Translation from Semi-extended Regular Expressions into NFA and Its Application to an Approximate Matching Problem, International Symposium on Algorithms and Computation, ISAAC'03, Kyoto, Japan, December 15-17, Lecture Notes in Computer Science 2906,158-167, Springer, Berlin, Heidelberg, 2003.
47. G. T. Thomason and P. N. Marinos, Deterministic acceptors of regular fuzzy languages, IEEE Transactions on Systems, Man, and Cybernetics, 4(2), 228-230, 1974.
48. Y. Cao, and Y. Ezawa, Nondeterministic fuzzy automata, Information Sciences, 191, 86-97, 2012.
49. P. Thiemann and M. Sulzmann, From ω-regular expressions to Büchi automata via partial derivatives, International Conference on Language and Automata Theory and Applications(LATA 2015), Nice, France, March 2-6, 2015.
50. D. Kuske and I. Meinecke, Construction of tree automata from regular expressions, International Conference on Developments in Language Theory, DLT 2008: Developments in Language Theory, 491-503, 2008.
51. S. Owens, J. Reppy and A. Turon, Regular-expression derivatives re-examined. Journal of Functional Programming, 19(2), 173-190, 2009.

52. E. Maia, N. Moreira and R. Reis, Partial derivative and position bisimilarity automata, International Conference on Implementation and Application of Automata (CIAA2014): Implementation and Application of Automata, 264-277, 2014.
53. J. M. Champarnaud and D. Ziadi, Canonical derivatives, partial derivatives and finite automaton constructions, Theoretical Computer Science, 289(1),137-163, 2002.
54. H. Chen, Derivatives of regular expressions, Technical report ISCAS-SKLCS-09-11, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, 2009.
55. A. Kumar and A. K. Verma, Conversion of parallel regular expressions to non-deterministic finite automata using partial derivatives, Chiang Mai Journal of Science, 41(5/2), 1409-1418, 2014.
56. V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, Theoretical Computer Science, 155, 291-319, 1996.
57. J. C. Martin, Introduction to Languages and the Theory of Computation, The McGraw-Hill Companies, New York, 1991.
58. D. Qiu, Characterizations of fuzzy finite automata, Fuzzy Sets and System, 141(3), 391-414, 2004.
59. E. T. Lee and L. A. Zadeh, Note on fuzzy languages, Information Sciences, 1(4), 421-434, 1969.
60. M. Sulzmann and P. Thiemann, Derivatives for Regular Shuffle Expressions. International Conference on Language and Automata Theory and Applications (LATA 2015), Nice, France, 2-6 March, 2015.
61. H. Gruber and M. Holzer, From Finite Automata to Regular Expressions and Back-A Summary on Descriptional Complexity, International Journal of Foundations of Computer Science, 26(8), 2015.
62. Y. Li, A categorical approach to lattice-valued fuzzy automata, Fuzzy sets and Systems, 157(6), 855-864, 2006.
63. M. Droste and P. Gastin, Weighted automata and weighted logics, Theoretical Computer Science, 380(1-2), 69-86, 2007.
64. P. R. J. Asveld, Algebraic aspects of families of fuzzy languages, Theoretical Computer Science, 293(2), 417-445, 2003.

65. M. Mizumoto, J. Toyoda and K. Tanaka, Examples of formal grammars with weights, Information Processing Letters, 2(3), 74-78, 1973.
66. M. Doostfatemeh and S. C. Kremer, New directions in fuzzy automata, International Journal of Approximate Reasoning, 38(2), 175-214, 2005.
67. J. Jin and Q. Li, Fuzzy grammar theory based on lattices, Journal Soft Computing - A Fusion of Foundations, Methodologies and Applications, 16(8), 1415-1426, 2012.
68. Y. B. Jun and J. Kavikumar, Bipolar fuzzy finite state machines, Bulletin of the Malaysian Mathematical Sciences Society, 34(1), 181-188, 2011.
69. J. Kavikumar, A. B. Khamis and R. Kandasamy, Fuzzy entire sequence spaces, International Journal of Mathematics and Mathematical Sciences, 1-9, 2007.
70. K. Thompson, Programming Techniques: Regular expression search algorithm, Communications of the ACM, 11 (6): 419-422, 1968.
71. S. Sippu and E. Soisalon-Soininen, Parsing Theory, Vol. I, Languages and Parsing, of EATCS Monographs on Theoretical Computer Science (Springer, Berlin), 1988.
72. G. Berry and R. Sethi, From regular expressions to deterministic automata, Theoretical Computer Science, 48, 117-126, 1986.
73. L. Ilie and S. Yu, Follow automata, Information and Computation, 186(1), 140-162, 2003.
74. A. Brüggemann-Klein, Regular expressions into finite automata, Theoretical Computer Science, 120, 197-213, 1993.
75. J. M. Sempere, On a class of Regular-Like Expressions for Linear Languages, Journal of Automata, Languages and Combinatorics, 5(3), 343-354, 2000.
76. J. M. Champarnaud, J. P. Dubernard, H. Jeanne and L. Mignot, Two-Sided Derivatives for Regular Expressions and for Hairpin Expressions, Fundamenta Informaticae, 137(4), 425-455, 2015.
77. A. Stamenković, M. Ćirić and J. Ignjatović, Reduction of fuzzy automata by means of fuzzy quasi-orders, Information Sciences, 275, 168-198, 2014.
78. R. K. Singh and A. Kumar, Conversion of Fuzzy Regular Expressions to Fuzzy Automata using the Follow Automata, ME dissertation, Thapar University, 2014.

79. M. Mohri, Generic ε -removal and input ε -normalization algorithms for weighted transducers, International Journal of Foundations of Computer Science, 13, 129-143, 2002.

80. C. Allauzen and M. Mohri, A Unified Construction of the Glushkov, Follow, and Antimirov Automata, International Symposium on Mathematical Foundations of Computer Science (MFCS 2006), StaráLesná, Slovakia, 110-121, August 28-September 2006.

81. J. J. Astrain, J. R. G. de Mendívil and J. R.Garitagoitia, Fuzzy automata with ε-moves compute fuzzy measures between strings, Fuzzy sets and systems, 157(11), 1550-1559, 2006.

82. K. M. Ravi and A. Choubey, Interval-valued fuzzy regular language, Journal of Applied Mathematics and Informatics, 28(3-4), 639-649, 2010.

83. S. R. Chaudhari and D. D. Komejwar, On fuzzy regular grammars, Advanced in Fuzzy Mathematics, 6(1), 89-103, 2011.

84. H. S. Lee, Minimizing fuzzy finite automata, Proceeding of 9th International Conference in Fuzzy Systems (FUZZ IEEE 2000), SanAntonio, TX, USA, 1, 65-70, 7-10 May 2000.

85. K. Peeva and Z. Zahariev, Computing behavior of finite fuzzy machines-Algorithm and its application to reduction and minimization, Information Sciences, 178(21), 4152-4165, 2008.

86. V. V. Topencharov and K. G. Peeva, Equivalence, reduction and minimization of finite fuzzy-automata, Journal of Mathematical Analysis and Applications, 84(1), 270-281, 1981.

87. K. Peeva, Equivalence, reduction and minimization of finite automata over semirings, Theoretical Computer Science, 88(2), 269-285, 1991.

88. W. Cheng and Z. W. Mo, Minimization algorithm of fuzzy finite automata, Fuzzy Sets and Systems, 141(3), 439-448, 2004.

89. C. W. Omlin, K. K. Thornber, C. L. Giles, Representation of fuzzy finite state automata in continuous recurrent neural networks, IEEE International Conference on Neural Networks, Washington DC, USA, 1023-1027, 3-6 June 1996.

90. A. Blanco, M. Delgado, M. C. Pegalajar, Fuzzy automaton induction using neural network, International Journal of Approximate Reasoning, 27, 1-26, 2001.

91. T. V. Le, Fuzzy finite automata and their application to speech recognition, Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems, IEEE, Gainesville, FL, USA, 269-272, 7-9 Dec, 1994.

92. V. K. Garg, Modeling of Distributed Systems by Concurrent Regular Expressions, Proceeding of the 2nd International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, Vancouver, BC, Canada, December 5-8, 313-327, 1989.

93. W. Gelade, W. Martens and F. Neven, Optimizing Schema Languages for XML: Numerical Constraints and Interleaving, SIAM Journal on Computing, 38(5), 2021-2043, 2009.

94. H. K. Ali and D. I. G. Amalarethinam, Activity Recognition with Multi-tape Fuzzy Finite Automata, International Journal of Modern Education and Computer Science, 5(5), 60-65, 2013.

95. S. Sharan and S. P. Tiwari, Products of rough finite state machines, National Conference on Computing and Communication Systems, 2012.

96. S. Sharan and S. P. Tiwari, Products of mealy-type rough finite state machines, National Conference on Computing and Communication Systems (NCCCS), IEEE, 1-5, Nov 21-22, 2012.

97. V. Karthikeyan and M. Rajasekar, Relation in fuzzy automata, Advances in Fuzzy Mathematics, 6(1), 121-126, 2011.

98. V. Karthikeyan and M. Rajasekar, γ -Synchronized fuzzy automata and their applications, **A**nnals of **F**uzzy **M**athematics and **I**nformatics, 10(2), 331-342, 2015.

99. Z. Yu, Fuzzy L languages, Fuzzy sets and Systems, 117(2), 317-321, 2001.

100. L. M. Reyneri, An introduction to fuzzy state automata, Biological and Artificial Computation: From Neuroscience to Technology, 273-283, 1997.

101. L. M. Reyneri and D. Morano, Fuzzy state automata and applications to mobile robot control, Intelligent Automation and Soft Computing, 11(2), 111-121, 2005.

102. S. Kovács, Fuzzy Reasoning and Fuzzy Automata in User Adaptive Systems, Proceedings of the 18th Hungarian-Korean Seminar on Soft Computing and Computational Intelligence, October 2-3, Budapest, Hungary, ISBN 963 206 0660, 63-72, 2002.

103. C. L. Giles, C. W. Omlin and K. K. Thornber, Deterministic encoding of fuzzy finite state automata in continuous recurrent neural networks, NEC Research Institute, Inc., U.S. Patent 5,943,659, 1999.
104. E. H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, In Proceedings of the Institution of Electrical Engineers, 121(12), 1585-1588, 1974.
105. K. Asai and S. Kitajima, A method for optimizing control of multimodal systems using fuzzy automata, Information Sciences, 3(4), 343-353, 1971.
106. S. K. De, R. Biswas and A. R. Roy, An application of intuitionistic fuzzy sets in medical diagnosis, Fuzzy sets and Systems, 117(2), 209-213, 2001.
107. F. Steimann and K. Adlassnig, Clinical monitoring with fuzzy automata, Fuzzy Sets and Systems, 61(1), 37-42, 1994.
108. F. Steimann, and K.P. Adlassnig, Fuzzy medical diagnosis, In E.H. Ruspini, P.P. Bonissone, W. Pedrycz (Eds.), Handbook of fuzzy computation, IOP Publishing Ltd./Oxford University Press, Bristol, UK, 1, 1-14, 1998.
109. M. G. Thomason, Finite fuzzy automata, regular fuzzy languages, and pattern recognition, Pattern Recognition, 5(4), 383-390, 1973.
110. X. Z. Gao and S. J. Ovaska, Motor Fault Detection and Diagnosis Using Soft Computing, Applied soft computing, 1(1), 73-81, 2001.
111. Q. E. Wu, X. M. Pang and Z. Y. Han, Fuzzy automata system with application to target recognition based on image processing, Computers and Mathematics with Applications, 61(5), 1267-1277, 2011.
112. A. Mateescu, A.Salomaa, K. Salomaa and S. Yu, Lexical analysis with a simple finite-fuzzy-automaton model, Journal of Universal Computer Science, 1(5), 292-311, 1995.
113. http://dinosaur.compilertools.net
114. M. Becchi, and P. Crowley, Extending finite automata to efficiently match perl-compatible regular expressions, In Proceedings of the 2008 ACM CoNEXT Conference (p. 25), Madrid, Spain, Dec 9-12, 2008.
115. S. Soni, P. Khanna and P. Tandon, Generative Evolutionary Action Grammar Based Form Exploration, 33rd Computers and Information in Engineering Conference, Portland, Oregon, USA, August 4-7, 2013.
116. P. R. J. Asveld, A fuzzy approach to erroneous inputs in context-free language recognition, Proceedings of the Fourth International Workshop on Parsing

Technologies (IWT 1995) September 20-24, Prague and Karlovy Vary, Czech Republic, 14-25, 1995.

117. P. R. J. Asveld, Fuzzy context-free languages Part 2: Recognition and parsing algorithms, Theoretical Computer Science, 347(1-2), 191-213, 2005.

118. M. Schneider, H. Lim and W. Shoaff, The utilization of fuzzy sets in the recognition of imperfect strings, Fuzzy Sets and Systems, 49 (3), 331-337, 1992.

119. M. Inui, W. Shoaff, L. Fausett and M. Schneider, The recognition of imperfect strings generated by fuzzy context-sensitive grammars, Fuzzy sets and Systems, 62(1), 21-29, 1994.

120. I. Bucurescu and A. Pascu, Fuzzy pushdown automata, International Journal of Computer Mathematics, 10(2), 109-119, 1981.

121. C. Moraga, An approach to fuzzy context-free languages, In Proceedings of XIV Spanish Congress on Fuzzy Logic Technologies, Langreo-Mires, 17-19 September, 73-78, 2008.

122. H. Xing, Fuzzy pushdown automata, Fuzzy Sets and Systems, 158(13), 1437-1449, 2007.

123. D. Arora, B. Hazela and V. Saxena, Semantics for UML model transformation and generation of regular grammar, ACM SIGSOFT Software Engineering Notes, 37(3), 1-5, 2012.

124. A. Hussain and M. Shabbir, Soft finite state machine, Journal of Intelligent and Fuzzy System, DOI: 10.3233/IFS-151642, 29, 1635-1641, 2015.

125. FLAT, http://search.cpan.org

126. A. Kumar, and A. K. Verma, An improved algorithm for the metamorphosis of semi-extended regular expressions to deterministic finite automata, The Computer Journal, 58(3), 448-456, 2015.

127. K. Singh, Conversion of deterministic finite automata to regular expression using bridge state, M.E. Thesis ,Thapar University, 2011.

128. S. Garhwal and R. Jiwari, Parallel Fuzzy Regular Expression and its Conversion to Epsilon-Free Fuzzy Automaton, The Computer Journal, 59(9), 1383-1391, 2016.

129. N. Kalra and A. Kumar, Fuzzy state grammar and fuzzy deep pushdown automaton, Journal of Intelligent & Fuzzy Systems, 31(1), 249-258, 2016.