CHAPTER 1

INTRODUCTORY OVERVIEW

This thesis is a study of extended fuzzy languages such as intuitionistic fuzzy language, interval-valued fuzzy language and vague language obtained using the generalizations of fuzzy set namely intuitionistic fuzzy sets, interval-valued fuzzy sets and vague sets respectively. Finite automata (DFA and NDFA) models with intuitionistic fuzzy transitions, interval-valued fuzzy transitions and vague transitions are proposed to recognize the above languages. Also, finite automata (DFA and NDFA) with intuitionistic fuzzy (final) states, interval-valued fuzzy (final) states and vague (final) states are proposed to process these languages. These languages are described by their respective regular expressions. Moreover, we have extended Myhill-Nerode theorem for intuitionistic fuzzy regular language, interval-valued fuzzy regular language and vague regular language. Finally, approximate string matching is performed using each model recognizing them and obtaining a relation between their membership values.

1.1 INTRODUCTION

Mathematical models in classical computation, automata have been an important area in theoretical computer science [1]. It started from a seminal paper of Kleene [2], and within a few years developed into a rich mathematical research topic. From the very beginning finite automata constituted a core of computer science. Part of the reason is that they capture something very fundamental as is witnessed by a numerous different characterizations of the family of rational languages, i.e. languages defined by finite automata [1]. In fact, the interrelation of finite automata and their applications in computer science is a splendid example of a really fruitful connection of theory and practice and these will accept regular language [3]. Finite automata played a crucial role in the theory of programming languages, compiler constructions, switching circuit designing, computer controller, neuron net, text editor and lexical analyzer [4].

Theoretical computer science uses models and analyses to study computers and computation [5]. It encompasses many areas of computer science to develop models and methods of analysis.
Automata theory is the study of abstract computing devices, or, “machines”. Before the computers, in the 1930’s, A. Turing studied an abstract machine that had all the capabilities of today’s computers, at least as far as in what they could compute. Turing’s goal was to describe precisely the boundary between what a computing machine could do and what it could not do; his conclusions apply not only to his abstract Turing machines, but on today’s real machines.

In the 1940’s and 1950’s, simpler kinds of machines, which we today call “finite automata”, were studied by a number of researchers. These automata, originally proposed to model brain function, turned out to be extremely useful for many other purposes, such as software for designing and checking the behavior of digital circuits, lexical analyzer, software for scanning large bodies of text, software for verifying systems of all types that have a finite number of distinct states and so on. Also in the late 1950’s the linguist N. Chomsky begun the study of formal “grammars”. These grammars have close relationships to abstract automata and serve today as the basis of some important software components.

All above theoretical developments bear directly on what computer scientists do today. Some of the concepts, like finite automata and certain kinds of formal grammars, are used in the design and construction of important kinds of software.

In this introductory chapter, we begin with briefing the important concepts of automata theory and cover the fuzzy sets with its three major extensions needed in our study in short.

### 1.2 THE CENTRAL CONCEPTS OF AUTOMATA THEORY

In this section we introduce the most important definitions that pervade the theory of automata. These concepts include the alphabet, strings, and languages.

**Definition 1.2.1:** An *alphabet* is a finite, nonempty set of symbols. It is denoted by $\Sigma$ and common alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c, ..., z\}$, the set of all lower-case letters.
3. The set of all ASCII characters or the set of all printable ASCII characters.
Definition 1.2.2: A string (or sometimes word) is a finite length sequence of symbols chosen from an alphabet. For example, 011 is a string from the binary alphabet $\Sigma = \{0,1\}$. The string 011011 is another string chosen from this alphabet.

The number of positions of symbols in the string is its length.

For instance, 011 have length 3. The standard notation for the length of a string $w$ is $|w|$.

For example, $|011011| = 6$.

Definition 1.2.3: The empty string, denoted $\epsilon$, is a special string with zero occurrences of symbols. i.e., $\epsilon$ is said to have zero length. This string may be chosen from any alphabet.

The set of all strings over an alphabet $\Sigma$ is usually denoted by $\Sigma^*$.

For instance,

$\Sigma^* = \{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ or $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$.

Sometimes, we wish to exclude the empty string from the set of strings.

The set of nonempty strings from alphabet $\Sigma$ is denoted by $\Sigma^+$.

Thus, two appropriate equivalences are:

1. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
2. $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Definition 1.2.4: Let $x$ and $y$ be two strings. Then $xy$ denotes the concatenation of $x$ and $y$, i.e., the string formed by making a copy of $x$ and following it by a copy of $y$.

Example 1.1: Let $x = 0111$ and $y = 011011$. Then $xy = 0111011011$ and $yx = 0110111011$. For any string $w$, the equations $\epsilon w = w \epsilon = w$ hold. i.e., $\epsilon$ is the identity for concatenation.

Definition 1.2.5: A set of strings all of which are chosen from some $\Sigma^*$, where $\Sigma$ a particular alphabet is called a language.

If $\Sigma$ is an alphabet, and $L \subseteq \Sigma^*$, then $L$ is a language over $\Sigma$.

Note that a language over $\Sigma$ need not include strings with all the symbols of $\Sigma$.

Some abstract examples are;
1. The set of binary numbers whose value is a prime: \{10, 11, 101, 111, \ldots\}.

2. \(\Sigma^*\) is a language for any alphabet \(\Sigma\).

3. \(\emptyset\), the empty language, is a language over any alphabet.

4. \(\{\varepsilon\}\), the language consisting of only the empty string, is also a language over any alphabet. Note that \(\emptyset \neq \{\varepsilon\}\); the former has no strings and the latter has one string.

The only important constraint on what can be a language is that all alphabets are finite. Thus languages, although they can have an infinite number of strings, are restricted to strings drawn from one fixed, finite alphabet.

### 1.3 Finite State Automata

In this section we shall introduce the definition of finite automata, its types and their relation, regular language and minimization of finite automaton. Finite automata played a crucial role in the theory of programming languages and compiler constructions [4]. Automata have been used to model the dynamics of Discrete Event Systems (DES). Such systems are usually met in manufacturing, database concurrency control, telecommunication or computer networks, and in biomedical applications such as clinical monitoring. The theory of finite automata can be used not only in computer science, but also in combinatorics and algebra. We also use finite automata in web search, in extraction of information from text to recognize set of keywords [5].

**Definition 1.3.1**: A **finite state automaton** is a mathematical pattern which has finite input and output systems.

Analytically, a **finite state automaton** (FSA) can be represented by a 5-tuple

\[ A = (Q, \Sigma, \delta, q_0, F), \]

where \(Q\) is a finite nonempty set of states; \(\Sigma\) is a finite nonempty set of inputs called input alphabet; \(\delta: Q \times \Sigma \rightarrow Q\) a function called transition function. This describes the change of states for each alphabet of \(\Sigma\) during the transition; \(q_0 \in Q\) is the initial state; and \(F \subseteq Q\) is the set of final states (it is assumed here that there may be more than one final state).

A string \(x \in \Sigma^*\) is accepted by a finite automaton \(A\) if \(\delta(q_0, x) = q\) for some \(q \in F\). The final state is also called an **accepting state**.
To make the notion of the language of an automaton precise, we need an extended transition function. It is the function that takes a state $q$ and a string $w$ and returns a state $q$. It is denoted by $\delta^*$ and is defined as $\delta^*(q, w) = \delta(\delta^*(q, x), a)$ for $w = xa \forall x \in \Sigma^*, a \in \Sigma$.

NOTE: An automaton in which the output depends only on the input is called an automaton without a memory. An automaton in which the output depends on the states also in addition to the input is called automaton with a finite memory. An automaton in which the output depends only on the states of the machine is called a Moore machine. An automaton in which the output depends on the state and the input at any instant is called a Mealy machine.

There are two types of finite automata namely deterministic and nondeterministic finite automata described briefly as below.

**Definition 1.3.2:** A deterministic finite automaton (DFA) is a “5-tuple” $A_1 = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is a finite nonempty set of states; $\Sigma$ is a finite nonempty set of inputs called input alphabet; $\delta : Q \times \Sigma \rightarrow Q$ a function called transition function. This takes as arguments a state and an input symbol and returns a state; $q_0 \in Q$ is the initial state; and $F \subseteq Q$ is the set of final states (it is assumed here that there may be more than one final state).

**Example 1.2:** Let $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $q_0 = \{q_0\}$, $q_1 \in F$, then, the transition diagram for the DFA accepting all strings with substrings 01 is as sketched below ($q_i$ represented by a double circle is the final or accepting state):

![Transition diagram for a DFA](image)

The transition table corresponding to the function $\delta$ of Example 1.3 is shown in Table 1.1. Here the starting state and the accepting states are marked with an arrow and a circle respectively.
Table 1.1: Transition table of Figure 1.1

<table>
<thead>
<tr>
<th>Q</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₀</td>
<td>q₂</td>
<td>q₀</td>
</tr>
<tr>
<td>q₁</td>
<td>q₁</td>
<td>q₁</td>
</tr>
<tr>
<td>q₂</td>
<td>q₂</td>
<td>q₁</td>
</tr>
</tbody>
</table>

The language of a DFA $A_1$ is denoted by $L(A_1)$, and is given as $L(A_1) = \{ w \mid \delta^*(q_0, w) \in F \}$.

That is, the language of $A_1$ is set of all strings $w$ that result in a sequence of state transitions from the starting state $q_0$ to one of the final (accepting) states (in terms of the transition diagram, it is the set of all labels along all the paths that lead from the starting state to any accepting state).

**Definition 1.3.3:** A *nondeterministic finite automaton* (NDFA) is a “5-tuple” $A_2 = (Q, \Sigma, \delta, q_0, F)$, where $Q$, $\Sigma$, $q_0$, $F$ are same as those in DFA and $\delta : Q \times \Sigma \rightarrow 2^Q$ a function called transition function. This takes a state in $Q$ and an input symbol in $\Sigma$ as arguments and returns a subset of $Q$.

Note that the only difference between a NDFA and a DFA is in terms of $\delta$ i.e., a set of states in the case of NDFA and a single state in the case of a DFA.

**Example 1.3:**

The NDFA in Figure 1.2 can be specified formally as $(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$, where the transition function $\delta$ is given by the following transition table.

![Figure 1.2: Transition diagram for NDFA](image)
Table 1.2: Transition table of Figure 1.2

<table>
<thead>
<tr>
<th>Q</th>
<th>Σ →</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₀</td>
<td>{q₀, q₁}</td>
<td>{q₀}</td>
<td></td>
</tr>
<tr>
<td>q₁</td>
<td>Φ</td>
<td>{q₂}</td>
<td></td>
</tr>
<tr>
<td>q₂</td>
<td>Φ</td>
<td>Φ</td>
<td></td>
</tr>
</tbody>
</table>

Note that transition tables can be used to specify the transition function for a NDFA as well as for a DFA. The only difference is that each entry in the table of NDFA is a set, even if the set is a singleton. Also, when there is no transition from a given state for the given input symbol, the proper entry is Φ, the empty set. States marked with an arrow and a circle respectively denote the starting state and the accepting states.

The language accepted by an NDFA \( A_2 \) is denoted by \( L(A_2) \), and is given by
\[
L(A_2) = \{ w \mid \delta^*(q_0, w) \cap F \neq \emptyset \}.
\]

That is, the language of \( A_2 \) is the set of all strings \( w \in \Sigma^* \) such that \( \delta^*(q_0, w) \) contains at least one accepting state.

**Equivalence of DFA & NDFA**

Although there are many languages for which a NDFA is easier to construct than a DFA, it is a surprising fact that every language described by some NDFA can also be described by some DFA. Moreover, the DFA in practice has about as many states as the NDFA, although it often has more transitions. In the worst case, however, the smallest DFA can have \( 2^n \) states while the smallest NDFA for the same language has only \( n \) states. After reading sequence of input symbols \( w \), the constructed DFA is in a state that is the set of states of NDFA. Since the accepting states of the DFA are those sets that include at least one accepting state of the NDFA we may conclude that the DFA and NDFA accepts exactly the same strings, and therefore accepts the same language.
Above description is proved in the following theorems.

**Theorem 1.1:** If \( D = (Q_D, \Sigma_D, \delta_D, \{q_0\}, F_D) \) is the DFA constructed from NDFA \( N = (Q_N, \Sigma_N, \delta_N, q_0, F_N) \) by the subset construction, then \( L(D) = L(N) \) [5].

**Theorem 1.2:** A language \( L \) is accepted by some DFA if and only if \( L \) is accepted by some NDFA [5].

### 1.4 FORMAL LANGUAGES AND FINITE AUTOMATA

The theory of formal languages is an area with a number of applications in Computer Science. Linguists were trying in the early 50s to define precisely valid sentences and give its structural descriptions. They wanted to describe the rules of grammar in a rigorous mathematical way to define a formal grammar. They thought such description of natural languages (languages like English, Hindi, etc we use in everyday life) would make language translation using computers easy. It was Noam Chomsky [5] who gave a mathematical model of a grammar in 1956. It turned out to be useful for computer languages, although it was not useful for describing natural languages such as English.

**Definition 1.4.1:** A *phrase-structure grammar* (or simply a *grammar*) [5] is a quadruple \( G = \{V_N, \Sigma, P, S\} \), where \( V_N \) is a finite nonempty set whose elements are called *variables* (non-terminals); \( \Sigma \) is a finite nonempty set whose elements are called *terminals*; \( V_N \cap \Sigma = \emptyset \); \( S \) is a special variable (\( S \in V_N \)) called the *starting symbol*; and \( P \) is a finite set whose elements are \( \alpha \rightarrow \beta \), where \( \alpha \) and \( \beta \) are strings on \( V_N \cup \Sigma \). \( \alpha \) has at least one symbol from \( V_N \). Elements of \( P \) are called *productions or rewriting rules or production rules*.

We consider the following two rules while writing production rules.

a) Reverse substitution is not permitted. i.e., if \( S \rightarrow AB \) is a production, then we can replace \( S \) by \( AB \), but we cannot replace \( AB \) by \( S \).

b) Inverse operation is not allowed. i.e., \( S \rightarrow AB \) is a production, it is not necessary that \( AB \rightarrow S \) is a production.
Derivations and the Languages Generated by a Grammar

**Definition 1.4.2:** If $\alpha \rightarrow \beta$ is a production in a grammar $G$ and $\gamma, \delta$ are any two strings on $V_N \cup \Sigma$, then we say $\gamma \alpha \delta$ directly derives $\gamma \beta \delta$ in $G$ and we write $\gamma \alpha \delta \xrightarrow{G} \gamma \beta \delta$. This process is called one-step derivation.

**Definition 1.4.3:** If $\alpha$ and $\beta$ are strings on $V_N \cup \Sigma$, then we say $\alpha$ derives $\beta$ if $\alpha \xrightarrow{*} \beta$. Here $\xrightarrow{*}$ denotes the reflexive-transitive closure of the relation $G \xrightarrow{}$ in $(V_N \cup \Sigma)^*.$

**Definition 1.4.4:** The language generated by a grammar $G$ is denoted by $L(G)$ and is defined by $L(G) = \{ w \mid w \in \Sigma^* \& S \xrightarrow{*} w \}$ or $L(G)$ contains the strings of terminal characters produced by applying the production rules of $G$. The elements of $L(G)$ are called sentences.

**Definition 1.4.5:** If $S \xrightarrow{*} \alpha$, then $\alpha$ is called a sentential form.

Note that elements of $L(G)$ are sentential forms but not vice versa.

**Definition 1.4.6:** If $L(G_1) = L(G_2)$, then $G_1 \equiv G_2$

**Remarks on Derivation**

a) Any derivation involves the application of productions. We write $\alpha \rightarrow \beta$ if the number of productions we apply is one and $\alpha \xrightarrow{*} \beta$ if it is more than one.

b) The string generated by the most recent application of production is termed as the working string.

c) A string is said to be derived completely when the working string cannot be modified. If the final string does not contain any variable, it is a sentence in the language and if it contains a variable, it is a sentential form and we say in this case the production gets ‘stuck.’

**Example 1.4:** Let $G = (\{ S, A_1 \}, \{ 0, 1, 2 \}, P, S)$, where $P$ consists of the productions

$S \rightarrow 0SA_1 2, S \rightarrow 012, 2A_1 \rightarrow A_1 2, 1A_1 \rightarrow 11$. Then $L(G) = \{ 0^n 1^n 2^n \mid n \geq 1 \}.$
Classification of Languages

We have \( G = \{V_N, \Sigma, P, S\} \) the definition of grammar, where \( V_N \) and \( \Sigma \) are the sets of symbols with \( S \in V_N \). Noam Chomsky classified the grammars into four types in terms of their types of productions (type 0 to type 3).

We need the following definition before entering the different types of productions.

**Definition 1.4.7:** In a production of the form \( \tau \rho \rightarrow \tau \alpha \rho \), where \( A \) is a variable, \( \tau \) is called the left context, \( \rho \) is right context, and \( \tau \alpha \rho \) the replacement string.

**Example 1.5:**

\( a) \) In \( lmABlmn \rightarrow lmABlmn \), \( lm \) is the left context, \( lmn \) is the right context, and \( \alpha = AB \).

\( b) \) In \( AC \rightarrow A \), \( A \) and \( \varepsilon \) are the left and right contexts respectively, and \( \alpha = \varepsilon \).

Here the production erases \( C \).

\( c) \) In \( A \rightarrow \varepsilon \), the left and right contexts are \( \varepsilon \), and \( \alpha = \varepsilon \). The production erases \( A \) in any context.

**Definition 1.4.8:** A production without any restrictions is called type 0 productions and a type 0 grammar is a phrase structure grammar having type 0 productions.

**Definition 1.4.9:** A production of the form \( \tau \rho \rightarrow \tau \alpha \rho \) is called type 1 if \( \alpha \neq \varepsilon \) (here erasing of \( A \) is not allowed).

For example, \( AC \rightarrow AcCb \) is a type 1 production.

A grammar is called type 1 or context-sensitive or context dependent if all its productions are of type 1, and the language generated by type 1 grammar is called a type 1 or context-sensitive language (\( L_{cd} \)).

**Definition 1.4.10:** A type 2 production is of the form \( A \rightarrow \alpha \), where \( A \in V_N \) and \( \alpha \in (V_N \cup \Sigma)^* \) (here L.H.S. has no left or right context).

For example, \( S \rightarrow ab, A \rightarrow bB \) are type 2 productions.
A grammar is said to be of *type 2 or context free*, if it contains only type 2 productions, and a language is called a *type 2 or context free language* ($L_{cfl}$) if it is generated by type 2 grammar.

**Definition 1.4.11:** A production of the form $B \rightarrow b, B \rightarrow aA$, where $A, B \in V_N, a, b \in \Sigma$ is called a *type 3* production.

A grammar is said to be of *type 3 or regular* if all its productions are of type 3 (here $S \rightarrow \varepsilon$ is permitted, but in this case $S$ does not appear on the right-hand side of any other production), and a language is *type 3 or regular language* ($L_{rl}$) if it is generated by type 3 grammar.

**Regular Expressions and Finite Automata**

Regular expressions are useful for representing certain sets of strings in an algebraic fashion. These describe languages (regular languages) accepted by finite state automata. The recursive definition of regular expressions over an alphabet $\Sigma$ is given as.

1. Any terminal symbol (i.e., an element of $\Sigma$), $\varepsilon$ and $\phi$ are regular expressions.
2. The union of two regular expressions $R_1$ and $R_2$ is also a regular expression. It is written as $R_1 + R_2$.
3. The concatenation of two regular expressions $R_1$ and $R_2$ is also a regular expression. It is written as $R_1R_2$.
4. The closure or iteration of a regular expression $R$, written as $R^*$, is also a regular expression.
5. If $R$ is a regular expression, then $(R)$ is also a regular expression, where $(R)$ denotes the order of evaluation of a regular expression.
6. The regular expressions over $\Sigma$ are precisely obtained recursively by the application of above rules once or several times.

Following theorems describes the relation between regular expression and finite automata and transition systems and regular expressions respectively.

**Theorem 1.3:** Every regular expression $R$ can be recognised by a transition system, i.e., for every string $w$ in the set $R$, there exists a path from the initial state to a final state with value $w$. 

11
Theorem 1.4: Any set L accepted by a finite automaton M can be represented by a regular expression.

NOTE:

1. It is possible to construct a finite automaton equivalent to a given regular expression, and to construct a regular expression equivalent to a given finite automata.
2. Two finite automata over $\Sigma$ are said to be equivalent if they accept/recognise the same set of strings over $\Sigma$.
3. Two regular expressions $P$ and $Q$ over $\Sigma$ are equivalent if and only if they represent the same set. Also, $P$ and $Q$ are equivalent if and only if the corresponding finite automata are equivalent.
4. We can show that the class of regular sets over $\Sigma$ are precisely regular languages over $\Sigma$. Also, if $G = (\{A_0, A_1, \ldots, A_n\}, \Sigma, P, A_0)$ is a regular grammar generating the regular language $L(G)$, then we can construct a finite automaton $A$ accepting $L$ whose:
   a. States correspond to variables.
   b. Initial state corresponds to $A_0$.
   c. Transitions correspond to the productions in $P$.

The necessary condition for an input string to be regular set is proved by theorem Pumping lemma. This theorem gives us a method of generating or pumping many input strings from a given string. The statement of the Pumping lemma is given below.

Theorem (Pumping lemma) 1.5: Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with $n$ states. Let $L$ be the regular set accepted by $A$. Let $w \in L$ and $|w| \geq n$. If $m \geq n$, then there exists $x, y, z$ such that $w = xyz$, $y \neq \varepsilon$ and $xy^iz \in L$ for each $i \geq 0[5]$.

1.5 MINIMIZATION OF FINITE AUTOMATA

One important result on finite automata, both theoretically and practically, is that for any regular language there is a unique DFA having the smallest number of states that accepts it.

The Myhill-Nerode theory is a branch of the algebraic theory of languages and automata in which formal languages and deterministic automata are studied through right invariant equivalence classes (also studied through right congruences and congruences on a free
monoid [5]). Myhill-Nerode theorem for finite automata is a very powerful tool for minimizing the number of states in a finite automaton. It provides necessary and sufficient conditions for a language to be regular, which are in terms of right invariant equivalence classes. However, the Myhill-Nerode theory not only deals with this theorem, but it also considers many other important topics. In particular, right congruences on a free monoid have shown oneself to be very useful in the proof of existence and construction of the minimal deterministic automaton recognizing a given language, as well as in minimization of deterministic automata. The statement of the theorem is as follows (proof of this theorem and an algorithm for minimizing a finite automaton is given in [5]).

**Theorem (Myhill-Nerode Theorem) 1.6:**

Following statements are equivalent.

i. \( L \) is a regular language.

ii. \( L \) is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.

iii. Define a relation \( R_L \) as \( xR_L y \) if and only if for all \( z \in \Sigma^*, xz \in L \) only when \( yz \in L \).

Then \( R_L \) is an equivalence relation of finite index.

1.6 FUZZY SETS

Among the various classical changes in science and mathematics in the previous century, one such change concerns the concepts of uncertainty. According to the traditional view, science should strive for certainty in all its manifestations hence; uncertainty (vagueness) is regarded as unscientific. According to modern view, uncertainty is considered essential to science; it is not only an unavoidable plague, but it has, in fact, a great utility. L. A. Zadeh in 1965 introduced the concept of “Fuzzy Sets” [6] to describe vagueness mathematically in its abstract form and tried to solve such problems by giving a grade of membership to each member of a given set. This in fact laid the foundations of fuzzy set theory. He stated that the “membership” in a fuzzy set is not a matter of affirmation or denial, but rather a matter of degree. Over the last forty years, his proposal has gained recognition as an important point in the evolution of modern concept of imprecision and uncertainty and his innovation represent a paradigm shift from the classical sets or the crisp sets to “Fuzzy Sets”. The fuzzy set theory
has a wider scope of applicability than classical set theory in solving various problems [7-12]. Fuzzy set theory in the last four decades has developed along two lines.

- as a formal theory formulated by generalizing the original ideas and concepts in classical mathematical areas.
- as a very powerful modelling language, that copes with a large fraction of uncertainties of real life situations.

Present-day science and technology is featured with complex processes and phenomena for which complete information is not always available. For such cases, mathematical models are developed to handle various types of systems containing elements of uncertainty. A large part of these models are based on a recent extension of the ordinary set theory, namely, the so-called fuzzy sets.

Fuzzy sets constitute one of computational intelligence’s most fundamental and influential tools. It has become one of the emerging areas in contemporary technologies of information processing. Recent studies in fuzzy set have spread across various areas such as, control, pattern recognition, and linguistic modelling. A significant number of direct real-world implementations range from home appliances to industrial installations involving fuzzy sets, both by themselves and hand in hand with other modern approaches, including neural networks. The concept of fuzzy sets is intellectually stimulating, and their applications are diverse and advanced [13-14].

In order to describe mathematically certain situations in which there is indeterminacy due to vagueness rather than randomness, Zadeh [6] introduced the concept of a fuzzy set in 1965 as an aggregate of elements with an ill-defined boundary. Fuzzy sets involve capturing, representing, and working with the concept of linguistics-objects without clear boundaries. What has been conveyed by fuzzy sets and formalized in the resulting framework is definitely essential to human deeds and spread through many studies on the role of uncertainty. Fuzzy sets emerged as a new way of representing uncertainty. As such, they naturally got involved in dedication of philosophical and methodological arguments with supporter of probability and statistics. It was determined quite early, however, that notions of randomness and fuzziness are mostly orthogonal and could eventually coexist. Fuzzy sets play a dominant role when it comes to building bridges between symbolic and numeric computation. These two areas constitute essential fortes of computing. The main idea behind fuzzy sets is the graded
distinction among elements of a universe of discourse. Fuzzy sets are formally defined as follows:

**Definition 1.6.1:** A fuzzy set [FS] is characterized by a membership (or fuzzy characteristic) function mapping the elements of a domain, space, or universe of discourse $X$ to the unit interval $[0, 1]$. i.e., $\mu_A : X \rightarrow [0,1]$.

The value of $\mu_A$ at $x$, $\mu_A(x)$, represents the grade of membership of $x$ in $A$. Thus, a fuzzy set $A$ in $X$ may be represented as a set of ordered pairs of a generic element $x \in X$ and its grade of membership: $\mu_A = \{(\mu_A(x)/x) \mid x \in X\}$. Clearly, a fuzzy set is a generalization of the concept of a set whose membership function takes on only two values {0, 1}, as discussed previously.

**Example 1.6:** Let $X$ be the set of integers from 0 to 100 representing the ages of individuals in a group, and let $A$ be a fuzzy set of a “middle-aged” individuals. Then, such a set may be characterized, subjectively of course, by the membership function given in the following table;

<table>
<thead>
<tr>
<th>$x$ (= age)</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>52</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_A(x)$</td>
<td>0.3</td>
<td>0.5</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.9</td>
<td>0.8</td>
<td>0.7</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

where the membership of the elements is assumed to be zero if not mentioned.

**Some Basic Operations on FSs:**

i) **Equality:** Two fuzzy sets $A$ and $B$ are equal, written as $A = B$, if and only if $\mu_A(x) = \mu_B(x)$ for all $x$ in $X$.

ii) **Containment:** A fuzzy set $A$ is contained in, or is a subset of a fuzzy set $B$, written as $A \subseteq B$, if and only if $\mu_A(x) \leq \mu_B(x)$ for all $x$ in $X$.

iii) **Complement:** The complement of a fuzzy set $A$ is denoted by $\overline{A}$ or $A^C$ and is defined by $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$ for all $x$ in $X$. i.e., $\overline{A}$ or $A^C \iff \mu_{\overline{A}}(x) = 1 - \mu_A(x)$, $\forall x \in X$. 
iv) Union: The union of two fuzzy sets $A$ and $B$, written as $A \cup B$, is a fuzzy set $C = A \cup B$ characterized by $\mu_C(x) = \max[\mu_A(x), \mu_B(x)]$, for all $x$ in $X$.

v) Intersection: The intersection of two fuzzy sets $A$ and $B$ is a fuzzy set $C$, written as $A \cap B$, defined by $\mu_C(x) = \min[\mu_A(x), \mu_B(x)]$, for all $x$ in $X$.

By a dual argument of the above, we see that $A \cap B$ is the largest fuzzy set which is contained in both $A$ and $B$.

Definition 1.6.2: A fuzzy set $A$ is empty if and only if it is identically zero on $X$. The empty fuzzy set will be denoted by $\phi$. i.e., $\phi \iff \mu_{\phi}(x) = 0$, $\forall x \in X$.

Definition 1.6.3: A fuzzy set $A$ is universal if and only if it is identically unit on $X$. The universal fuzzy set is a space $X$. i.e., $X \iff \mu_{X}(x) = 1$, $\forall x \in X$.

Definition 1.6.4: Given a fuzzy set $A$ in a finite universe $X$, its cardinality, denoted by $\text{Card}(A)$, is defined as $\text{Card}(A) = \sum_{x \in X} A(x)$.

Often, $\text{Card}(A)$ is referred to as the scalar cardinality or the sigma count of $A$.

For example, the fuzzy set $A = 0.1/1 + 0.4/2 + 0.5/3 + 0.7/4 + 1.0/5$ in $X = \{1, 2, 3, 4, 5, 6\}$ is such that $\text{Card}(A) = 2.7$.

The basic algebraic properties on fuzzy sets are given in [9]. From the properties concerning with fuzzy sets, we say they form a distributive lattice, but do not form a Boolean lattice, because $\overline{A}$ is not the complement of $A$ in the lattice sense.

1.7 GENERALIZATIONS OF FUZZY SET

Zadeh has defined a fuzzy set as a generalization of the characteristic function of a subset. A fuzzy set ‘$A$’ in ‘$U$’ the universe of discourse under discussion is identified by a membership function $\mu_A : U \rightarrow [0, 1]$ defined such that for any element $x$ in ‘$U$,’ $\mu_A(x)$ is a real number in the closed interval [0, 1] indicating the degree of membership of $x$ in ‘$A$’ [6]. Nearer the value of an element to unity, higher the grade of its membership. Intuitively, a fuzzy set ‘$A$’ has an unclear, ill-defined boundary so that an element $x$ is not necessarily either "in $A$" with $\mu_A(x) = 1$ or "not in $A"$ with $\mu_A(x) = 0$; rather, $x$ may have partial membership in ‘$A$’ with $0 < \mu_A(x) < 1$. The membership function $\mu_A$ can be viewed as an arithmetization which reflects the ambiguity of set ‘$A$.’ Since this single number does not tell us the
uncertainty/impreciseness completely; we need to generalize the membership function. As a result we come across many extensions of fuzzy sets and among them we have considered intuitionistic fuzzy set (IFS) [15], interval-valued fuzzy set (IVFS) [16-18], and vague set (VS) [19]. IFSs, IVFSs, and VSs are three intuitively straightforward extensions of Zadeh’s fuzzy sets that were considered independently to improve the preciseness of the belongingness of an element to a set.

A number of distinct generalizations have been made from the generic concept of partial membership conveyed by fuzzy sets. Although some of these extensions are primarily mathematical, others arise as an effect of difficulties in representing the concept of membership through single numerical values. The three extended fuzzy sets below fall under second category, conveying application-oriented constructs. Their role in application of fuzzy sets will be explained through several specific constructs, particularly in fuzzy modelling.

### 1.7.1 INTUITIONISTIC FUZZY SETS

In conventional fuzzy set, a membership function assigns to each element of the universe of discourse a number from the unit interval to indicate the degree of belongingness to the set under consideration. Since Zadeh introduced fuzzy sets in 1965, many new approaches and theories treating imprecision and uncertainty have been proposed. In 1986, Atanassov introduced the concept of an intuitionistic fuzzy set which is characterized by two functions expressing the degree of belongingness and the degree of non-belongingness, respectively. This idea, which is a natural generalization of a standard fuzzy set, seems to be useful in modelling many real life situations, like negotiation processes, etc.

The late George Gargov is the “god father” of the intuitionistic fuzzy sets in fact, he has invented the name “intuitionistic fuzzy,” motivated by the fact that the law of the excluded middle does not hold for them. Atanassov’s sets gave a very natural tool for modelling preferences. Sometimes it seems to be more natural to describe imprecise and uncertain opinions not only by membership functions. It is due to the fact that in some situations it is easier to describe our negative feelings than the positive attitude. Even more, quite often one can easily specify objects or alternatives one dislikes, but simultaneously cannot specify clearly what he really wants.
IFS theory basically defies the claim that an element \( x \) “belongs” to a given degree (say \( \mu \)) to a fuzzy set \( A \), naturally follows that \( x \) should “not belong” to \( A \) to the extent \( 1-\mu \), an assertion implicit in the concept of a fuzzy set. On the contrary, IFSs assign to each element of the universe both a degree of membership \( \mu \) and degree of non-membership \( \gamma \) such that \( 0 \leq \mu + \gamma \leq 1 \), thus relaxing the enforced duality \( \gamma = 1-\mu \) from fuzzy set theory. Obviously, when \( \mu + \gamma = 1 \) for all elements of the universe, the traditional fuzzy set concept is recovered. IFSs owe their name to the fact that this latter identity is weakened into an inequality, in other words: a denial of the law of the excluded middle occurs, one of the main ideas of intuitionism. Since then a great number of theoretical and practical results appeared in the area of intuitionistic fuzzy sets [20-28].

Let us consider a situation observed in a real estate agency. Very often a customer looking for an apartment is not convinced completely on the location and considers several variants. It is obvious that some districts are more preferable than others, when there are also districts that customer dislikes. It seems that intuitionistic fuzzy sets are very useful for modelling situations like this. For example, it may happen that a person asked about his favourite district in Karnataka cannot definitely choose whether it is Mysore, Bangalore, or Coorg, but he feels sure that he hates Hassan. Thus we may apply an intuitionistic fuzzy set for modelling the preferences of a customer, where the membership function shows the degree that a given district is the most preferred one, while the non-membership function indicates the degree that a given district should not be taken into consideration.

The definition of intuitionistic fuzzy set is given below.

Let a (non-fuzzy) set \( X \) be fixed.

**Definition 1.7.1.1:** An *intuitionistic fuzzy set* (IFS) \( A \) in \( X \) is defined as an object of the form

\[
A = \{(x, \mu_A(x), \gamma_A(x)) | x \in X\},
\]

where the functions:

\[
\mu_A(x) : X \to [0, 1] \quad \text{and} \quad \gamma_A(x) : X \to [0, 1]
\]

define the degree of membership and the degree of non-membership of the element \( x \in X \), respectively, and for every \( x \in X : 0 \leq \mu_A(x) + \gamma_A(x) \leq 1 \) for all \( x \in X \).

Obviously, each ordinary fuzzy set may be written as \( \{(x, \mu_A(x), 1-\mu_A(x)) | x \in X\} \).
The value of \( \pi_A(x) = 1 - \mu_A(x) - \gamma_A(x) \) is called the degree of non-determinacy (uncertainty or intuitionistic fuzzy index or the hesitation margin) of the element \( x \in X \) to the IFS \( A \).

**Definition 1.7.1.2:** Two IFSs \( A \) and \( B \) are equal i.e., \( A = B \) if and only if \( A \leq B \) and \( B \leq A \).

**Definition 1.7.1.3:** An IFS set ‘\( A \)’ is the subset of ‘\( B \)’

i.e., \( A \leq B \) iff \( \mu_A(x) \leq \mu_B(x) \) and \( \gamma_A(x) \geq \gamma_B(x) \) \( \forall x \in X \).

**Some Algebraic Operations over IFSs:**

Here, we mention some of the basic operations on IFS’s [22].

If \( A \) and \( B \) are two IFS’s over the universe of discourse \( X \), then

**Union:** \( A \cup B = \{(x, \max(\mu_A(x), \mu_B(x)), \min(\gamma_A(x), \gamma_B(x)))| x \in X \} \).

**Intersection:** \( A \cap B = \{(x, \min(\mu_A(x), \mu_B(x)), \max(\gamma_A(x), \gamma_B(x)))| x \in X \} \).

**Complement:** \( \overline{A} \) or \( A^C = \{(x, \gamma_A(x), \mu_A(x))| x \in X \} \).

Similarly, we can define some more algebraic properties on IFSs.

IFSs as a generalization of fuzzy sets can be useful in situations when description of a problem by a (fuzzy) linguistic variable, given in terms of a membership function only, seems too rough. For example, in decision making problems, particularly in the case of medical diagnosis, sales analysis, new product marketing, financial services, etc. there is a fair chance of the existence of a non-null hesitation part at each moment of evaluation of an unknown object. IFS are useful in modelling many real life situations, like negotiation processes. Like all young theories, the theory of IFSs contains a lot of open problems. While well established theories contain famous problems with solutions that seem a matter of distant future, a number of “technical” problems persist in new theories, perhaps not so hard but requiring plenty of time and research effort [27-28].

**1.7.2 INTERVAL-VALUED FUZZY SETS**

The theory of fuzzy set, pioneered by Zadeh [6], has achieved many successful applications in practice. As a rule, the membership functions of fuzzy sets representing particular verbal
expressions cannot be defined unequivocally on the basis of available information. Therefore, it is not always possible for a membership function of the type \( \mu : X \to [0, 1] \) to assign precisely one point from the interval \([0, 1]\) to each element \( x \in X \) without the loss of at least a part of information. As a generalization of fuzzy set, Zadeh [16], introduced the concept of interval-valued fuzzy set. Further many authors investigated the topic and obtained some meaningful conclusions. Gorzalczyzny [29] have applied interval-valued fuzzy sets in approximate reasoning. This work shows the importance of interval-valued fuzzy sets.

Interval-valued fuzzy set theory emerged from the observation that in many cases, no objective procedure is available to select the crisp membership degrees of elements in a fuzzy set. It was suggested to lighten that problem by allowing an interval \([\mu_1, \mu_2]\) to which the actual membership degree is assumed to belong. A related approach, second-order fuzzy set theory, also introduced by Zadeh, goes one step further by allowing the membership degrees themselves to be fuzzy sets in the unit interval; this extension is not considered in this thesis. The theory and applications (some) of interval-valued fuzzy sets has been developed in many areas so far as given in [30-37].

Most of the fuzzy modelling efforts are assuming a membership function, which could be regarded as a point estimate of the degree of belief of belongingness relation, for the reflection of vague nature of system. However, it may be more logical and practical to assume an interval-valued membership grade, which could be regarded as an interval-valued estimate of the degree of belief of the subordination relation because as a general and natural human thinking pattern, the degree of fuzziness appears as an interval-valued number on \([0, 1]\).

In interval-valued fuzzy set theory, the interval \([0, 1]\) is replaced by the set of subintervals of \([0, 1]\). Since an interval is completely determined by its endpoints, this set of intervals can be identified with the set \(\{(a, b) : a, b \in [0, 1], a \leq b\}\). The element \((a, b)\) is just a pair with \(a \leq b\). We will refer these pairs as intervals. In lattice theory the standard notation for this set is \([0,1]^2\). So an interval-valued fuzzy subset of a set \(S\) in lattice theory is given by the mapping \(A : S \to [0, 1]^2\). Here, the construction of the membership function is made using intervals in the fuzzy membership. So, instead of treating membership value as a real number in the image, the membership function defined for the fuzzy interval sets will work with an interval of real numbers in the interval \([0, 1]\), or will be sub-intervals of the interval \([0, 1]\) in the image. The definition of interval-valued fuzzy set is given in definition 1.7.2.1.
Definition 1.7.2.1: Let ‘U’ denote a universe of discourse. Let \([0, 1]\) denote the set of all closed subintervals of the interval \([0, 1]\).

An interval-valued fuzzy set (IVFS) [32] ‘A’ is a mapping \(\mu_A : U \rightarrow I[0, 1]\).

For all \(u \in U\),
\[
\mu_A(u) = [\mu_A^L(u), \mu_A^U(u)],
\]
where \(\mu_A^L(u), \mu_A^U(u) : U \rightarrow [0, 1]\) represents, the lower and the upper membership values of each element \(u \in U\) in ‘A’ such that \(0 \leq \mu_A^L(u) \leq \mu_A^U(u) \leq 1, \forall u \in U\).

Some Algebraic Operations over IVFSs:

For given interval-valued fuzzy sets \(A, B \subseteq X\) the universe of discourse, the union, intersection, complement and cross product are defined as following:

i) \(A \cup B = \{(x, \mu_{A \cup B}(x)) \mid x \in X\}\), where \(\mu_{A \cup B}(x) = [\mu_{A \cup B}^L(x), \mu_{A \cup B}^U(x)]\), \(\forall x \in X\) for
\[
\mu_{A \cup B}^L(x) = [\mu_A^L(x) \lor \mu_B^L(x)], \quad \mu_{A \cup B}^U(x) = [\mu_A^U(x) \lor \mu_B^U(x)].
\]

ii) \(A \cap B = \{(x, \mu_{A \cap B}(x)) \mid x \in X\}\), where \(\mu_{A \cap B}(x) = [\mu_{A \cap B}^L(x), \mu_{A \cap B}^U(x)]\), \(\forall x \in X\) for
\[
\mu_{A \cap B}^L(x) = [\mu_A^L(x) \land \mu_B^L(x)], \quad \mu_{A \cap B}^U(x) = [\mu_A^U(x) \land \mu_B^U(x)].
\]

iii) The complement of an interval-valued fuzzy set \(A \subseteq X\) is given by
\[
\overline{A} \text{ or } A^C = 1 - \mu_A(x) = \{(x, [1 - \mu_A^L(x), 1 - \mu_A^U(x)])\}.
\]

iv) For \(A \subseteq X\), and \(B \subseteq Y\), \(A \times B\) is defined as
\[
A \times B = \{((x, y), \mu_{A \times B}(x, y)) \mid x \in X, y \in Y\},
\]
where \(\mu_{A \times B}(x, y) = [\mu_{A \times B}^L(x, y), \mu_{A \times B}^U(x, y)]\) for
\[
\mu_{A \times B}^L(x, y) = \mu_A^L(x) \land \mu_B^L(y), \quad \mu_{A \times B}^U(x, y) = \mu_A^U(x) \land \mu_B^U(y),
\]
where \(\lor\) means maximum, and \(\land\) means minimum.

Definition 1.7.2.2: An interval-valued fuzzy set ‘A’ is the subset of ‘B’ i.e.,
\(A \subseteq B\) if \(\mu_A^L(x) \leq \mu_B^L(x)\) and \(\mu_A^U(x) \leq \mu_B^U(x)\), and

Definition 1.7.2.3: Two interval-valued fuzzy sets ‘A’ and ‘B’ are equal i.e.,
\(A = B\) if \(\mu_A^L(x) = \mu_B^L(x)\) and \(\mu_A^U(x) = \mu_B^U(x)\).
IVFS can be applied to study inference in approximate reasoning, representing default reasoning, decision making, etc [36].

1.7.3 VAGUE SETS

Vague sets [19] came into picture due to the concept of membership value of an element of fuzzy set. It says that the single entry combines the evidence for \( x \in X \) and the evidence against \( x \in X \) without indicating how much of each is there and thus tells us nothing about its accuracy. To overcome this difficulty, vague set allows the membership value in a continuous interval of real numbers in the range \([0, 1]\). This subinterval keeps track of both the favouring evidence and the opposing evidence respectively called the truth membership function \( t_A(x) \) and false membership function \( f_A(x) \) to record the lower bounds on \( \mu_A(x) \). These lower bounds are used to create a subinterval of \([0, 1]\), namely \([t_A(x), 1-f_A(x)]\), to generalize the \( \mu_A(x) \) of fuzzy sets. The lower bound and upper bound of this subinterval are \( t_A(x) \) and \( 1-f_A(x) \), respectively.

**Definition 1.7.3.1:** Let ‘\( U \)’ denote a universe of discourse. A vague set (VS) [19] ‘\( A \)’ in ‘\( U \)’ is characterized by a truth membership function \( t_A \) and a false membership function \( f_A \).

\( t_A(x) \) is a lower bound on the grade of membership of \( x \) derived from the evidence for \( x \), and \( f_A(x) \) is a lower bound on the negation of \( x \) derived from the evidence against \( x \). \( t_A(x) \) and \( f_A(x) \) both associate a real number in the interval \([0, 1]\) with each point \( x \) in ‘\( U \)’, where \( 0 \leq t_A(x) + f_A(x) \leq 1 \) i.e., \( t_A : U \rightarrow [0, 1] \) and \( f_A : U \rightarrow [0, 1] \).

This approach bounds the grade of membership of \( x \) to a subinterval \([t_A(x), 1-f_A(x)]\) of \([0, 1]\).

Precisely, the grade of membership \( \mu_A(x) \) of \( x \) may be unknown, but is bounded by

\[
0 \leq t_A(x) \leq \mu_A(x) \leq 1 - f_A(x)
\]

where \( 0 \leq t_A(x) \leq f_A(x) \leq 1 \).

**Definition 1.7.3.2:** A vague set ‘\( A \)’ is empty if and only if its truth-membership and false-membership functions are identically zero on ‘\( U \)’ the universal set.

**Definition 1.7.3.3:** A vague set ‘\( A \)’ is contained in a vague set ‘\( B \)’ denoted by \( A \subseteq B \), iff \( t_A(x) \leq t_B(x), 1-f_A(x) \leq 1-f_B(x) \).
**Definition 1.7.3.4:** Two vague sets ‘A’ and ‘B’ are equal, written as \( A = B \), iff \( A \subseteq B \), and \( B \subseteq A \); i.e., \( t_A(x) = t_B(x) \), \( 1 - f_A(x) = 1 - f_B(x) \).

**Some Algebraic Operations over VSs:**

i) **Union:** The union of two vague sets ‘A’ and ‘B’ is a vague set \( C \), written \( C = A \cup B \), where \( t_C = \max(t_A, t_B) \), \( 1 - f_C = \max(1 - f_A, 1 - f_B) = 1 - \min(f_A, f_B) \).

ii) **Intersection:** The intersection of two vague sets ‘A’ and ‘B’ is a vague set \( C \), written \( C = A \cap B \), where \( t_C = \min(t_A, t_B) \), \( 1 - f_C = \min(1 - f_A, 1 - f_B) = 1 - \max(f_A, f_B) \).

Here, \( t_A, f_A, t_B \) and \( f_B \) respectively denotes the truth-membership and false-membership functions ‘A’ and ‘B’.

iii) **Complement:** The complement of a vague set ‘A’ is denoted by \( \overline{A} \) and is defined by \( t_{\overline{A}}(x) = f_A(x) \), \( 1 - f_{\overline{A}}(x) = 1 - t_A(x) \).

**1.7.4 RELATION BETWEEN IFS, IVFS, & VS**

All three approaches, IFS, IVFS, and VS theory, have the virtue of complementing fuzzy sets, with ability to model uncertainty as well. IVFSs reflect this uncertainty by the length of the interval membership degree \([\mu_1, \mu_2]\), while in IFS theory for every membership degree \((\mu, \gamma)\), the value \( \pi = 1 - \mu - \gamma \) denotes a measure of non-determinacy (or undecidedness), and in VS theory, the subinterval \([t_A(x), 1 - f_A(x)]\) of \([0, 1]\) generalize the membership value \( \mu_A(x) \) of fuzzy sets. Each approach has given rise to an extensive literature covering their respective applications, but surprisingly very few people seem to be aware of their equivalence, stated in [38-40]. Indeed, take any IVFS \( A \) in a universe \( X \), and assume that the membership degree of \( x \) in \( A \) is given as the interval \([\mu_1, \mu_2]\). Obviously, \( \mu_1 + 1 - \mu_2 \leq 1 \), so by defining \( \mu = \mu_1 \) and \( \gamma = 1 - \mu_2 \) we obtain a valid membership and non-membership degree for \( x \) in an IFS \( A' \). Conversely, starting from any IFS \( A' \) we may associate to it an IVFS \( A \) by assigning, for each element \( x \), the membership degree of \( x \) in \( A \) equal to the interval \([\mu, 1 - \gamma]\) with again \( (\mu, \gamma) \) the pair of membership/non-membership degrees of \( x \) in \( A' \)[41-44]. Also, the membership degree and non-membership degree of an element \( x \) in an IFS can be written as a truth membership value and false membership value of an element \( y \) respectively in a VS.
As a consequence, a considerable work has been duplicated by adepts of either theory, or worse, is known to one group and ignored by the other. Therefore, regardless of the meaning (semantics) that one likes his or her preferred approach to convey, it is worthwhile to develop the underlying theory in a framework as abstract and general as possible.

1.8 FUZZY MODELLING

Fuzzy modelling undoubtedly becomes vital whenever any application of fuzzy sets is anticipated. The advancements in modelling translate directly into more advanced methodology, better understanding of fundamental concepts of fuzzy sets, and design practices of user-friendly models. Because mathematical and experimental components obviously coexist in fuzzy sets, the development of a sound modelling methodology becomes a must.

Briefly speaking, fuzzy models are modelling constructs featuring two main properties:

- They operate at the level of linguistic terms (fuzzy sets); similarly all system dependencies can be portrayed in the same linguistic format,
- They represent and process uncertainty.

We can enumerate several general types of fuzzy models. The proposed taxonomy depends on the level of structural dependencies visualised among the system’s variables and captured by the specific model. Among the fuzzy models, we proceed with fuzzy grammars (other classes of fuzzy models include Tabular representations, Fuzzy-relational equations, Fuzzy neural networks, Rule-based models, Local regression models, and Fuzzy regression models. These concepts are not discussed in this thesis).

1.8.1 FUZZY GRAMMARS

The notion of fuzzy grammars defined by Lee and Zadeh [46], and Mizumoto, et al. [47], Kandel [48], Novaks [49] is a natural generalization of the definition of formal grammars. Fuzzy grammars and fuzzy languages are fuzzy symbol-oriented formalisms that can be readily used in the description of various systems. They are of particular interest in characterizing time series and developing signal classifiers.

**Definition 1.8.1.1:** A fuzzy grammar [46] is defined as a quadruple \( G = (V_N, V_T, P, S) \), where
$V_T$ is the finite set of alphabets called terminal symbols, $V_N$ is the finite set of non-terminal symbols $(V_N \cap V_T = \phi)$, $P$ is a list of production (rewrite) rules, and $S$ is an initial symbol included in $V_N$.

The elements of $P$ define conditioned fuzzy sets in $(V_N \cup V_T)^*$ (A fuzzy set conditioned on $\alpha$ is a fuzzy set whose membership function depends on $\alpha$ as a parameter). The elements of $P$ are expressions of the form $\mu(\alpha \rightarrow \beta) = \theta, \theta > 0$, where $\alpha$ and $\beta$ are strings in $(V_N \cup V_T)^*$ and $\theta$ is the grade of membership of $\beta$ for a given $\alpha$ [46].

For convenience we abbreviate $\mu(\alpha \rightarrow \beta) = \theta$ to $\alpha \overset{\theta}{\rightarrow} \beta$ or, more simply, $\alpha \rightarrow \beta$.

As in the case of non-fuzzy grammars, the expression $\alpha \rightarrow \beta$ represents a rewriting rule. Thus, if $\alpha \overset{\theta}{\rightarrow} \beta$ and $\gamma$ and $\delta$ are arbitrary strings in $(V_N \cup V_T)^*$, then $\gamma \alpha \delta \rightarrow \theta \gamma \beta \delta$ means $\gamma \beta \delta$ is directly derivable from $\gamma \alpha \delta$.

If $\alpha_1, \alpha_2, \ldots, \alpha_m$ are strings in $(V_N \cup V_T)^*$ and $\alpha_1 \overset{\theta_1}{\rightarrow} \alpha_2, \ldots, \alpha_{m-1} \overset{\theta_m}{\rightarrow} \alpha_m, \theta_1, \ldots, \theta_m > 0$, then $\alpha_1$ is said to derive $\alpha_m$ in the grammar in $G$, and is expressed by $\alpha_1 \Rightarrow \alpha_m$.

**Types of Fuzzy Grammars**

Paralleling the standard classification of grammars, we can distinguish four principal types of fuzzy grammars as enlisted below [46].

**Type 0 fuzzy grammar:** In this case, the productions are of the general form $\alpha \overset{\theta}{\rightarrow} \beta, \theta > 0$, where $\alpha$ and $\beta$ are strings in $(V_N \cup V_T)^*$.

**Type 1 fuzzy grammar or context-sensitive fuzzy grammar (CSFG):** Here the productions are of the form $\alpha_1 A \alpha_2 \overset{\theta}{\rightarrow} \alpha_1 B \alpha_2, \theta > 0$, with $\alpha_1, \alpha_2$, and $B$ as strings in $(V_N \cup V_T)^*$, $A$ in $V_N$, and $B \neq \varepsilon$ (In addition, the production $S \rightarrow \varepsilon$ is allowed).

**Type 2 fuzzy grammar or context-free fuzzy grammar (CFFG):** The allowable productions in this grammar are of the form $A \overset{\theta}{\rightarrow} B, \theta > 0$, $A \in V_N$, $B \in (V_T \cup V_N)^*$, $B \neq \varepsilon$, and $S \rightarrow \varepsilon$.

**Type 3 fuzzy grammar or regular fuzzy grammar (RFG):** In this case we allow the following productions (including production $S \rightarrow \varepsilon$);
A\rightarrow^{\theta}aB \text{ or } A\rightarrow^{\theta}a, \theta > 0, A, B \text{ in } V_N, a \text{ in } V_T, 0 \leq \theta \leq 1.

It is possible to define an n-fold fuzzy grammar in which the grade of the application of the rewriting rule to be used next is conditioned by the n rules used before in a derivation, where \( n \geq 1 \). This generates an n-fold fuzzy language denoted by \( L(n - FG) \). Though fuzzy grammars are not a perfect model of natural languages, they are a good model since they reflect the uncertainty and inexactness involved in natural languages. Fuzzy grammars should not be confused with probabilistic grammars, although both formalisms are instances of the more general notion of so-called weighted grammars [50-52], i.e., grammars in which the production rules have been provided with weights. Fuzzy grammars have been found to be useful in a variety of applications such as in the analysis of X-rays [53], in digital circuit design, and in the design of intelligent human-computer interfaces. Also, it is possible to apply fuzzy grammars with time variant fuzzy vectors to characterize language learning systems.

1.8.2 FUZZY LANGUAGE

Natural languages are intrinsically imprecise. This contrast rather sharply with the precision of programming languages. In order to reduce the gap between them, it is natural to include randomness into the structure. This leads to the concept of stochastic languages. Another possibility lies in the introduction of fuzziness. This results in fuzzy language [54-61]. Fuzzy regular language is a feature of fuzzy language and is described by fuzzy regular expressions [62]. A fuzzy language is generated by a fuzzy grammar, the natural generalization of formal grammar which is introduced to reduce the gap between formal language and natural language (another way of reducing their gap is by introducing randomness [50-52]). Fuzzy languages and fuzzy grammars were introduced by Lee and Zadeh [46].

A fuzzy language \( \tilde{L} \), in \( \Sigma \) is a class of strings \( w \in \Sigma^* \) along with a grade of membership function \( \mu_{\tilde{L}}(w) \). This membership function \( \mu_{\tilde{L}}(w), w \in \Sigma^*, \) assigns to each string a grade of membership in \([0, 1]\). This single value combines the evidence for \( w \in \Sigma^* \) and the evidence against \( w \in \Sigma^* \), without indicating how much of each is there. This single number tells us nothing about its accuracy. To overcome this difficulty, we need to generalize the grade of membership function \( \mu_{\tilde{L}}(w) \) of fuzzy languages.
**Definition 1.8.2.1:** A fuzzy language \( \tilde{L} \) over an alphabet \( \Sigma \) is defined to be a fuzzy set in \( \Sigma^* \). \( \tilde{L} \) can be written as the set of ordered pairs
\[
\tilde{L} = \{(x, \mu_{\tilde{L}}(x)) \mid x \in \Sigma^* \},
\]
where \( \mu_{\tilde{L}}(x) \) is the grade of membership of \( x \) in \( \tilde{L} \).

We assume that \( \mu_{\tilde{L}}(x) \) is a number in the interval \([0, 1]\).

A trivial example of a fuzzy language is the set
\[
\tilde{L} = \{(a, 1.0), (b, 0.8), (ab, 0.7), (ba, 0.5), (bb, 0.6), (bb, 0.5)\} \text{ in } \{a, b\}^* \text{, where } \Sigma = \{a, b\}.
\]

It is understood that all strings in \( \{a, b\}^* \) other than those listed have the grade of membership 0 in \( \tilde{L} \).

A fuzzy grammar \( G \) generates a fuzzy language \( \tilde{L}(G) \) (FL) in the following manner.

A string of terminals \( x \) is said to be in \( \tilde{L}(G) \) if and only if \( x \) is derivable from \( S \) the starting symbol in \( G \). The grade of membership of \( x \) in \( \tilde{L}(G) \) is given by
\[
\mu_G(x) = \max(\min(\mu(S \rightarrow \alpha_1), \mu(\alpha_1 \rightarrow \alpha_2), ..., \mu(\alpha_m \rightarrow x))),
\]
where \( \mu_G(x) \) is an abbreviation for \( \mu_{\tilde{L}(G)}(x) \) and the maximum is taken over all derivation chains from \( S \) to \( x \). Thus, \( \mu_G(x) \) defines \( \tilde{L}(G) \) as a fuzzy set in \( (V_T)^* \). If \( \tilde{L}(G_1) = \tilde{L}(G_2) \) in the sense of equality of fuzzy sets, then the fuzzy grammars \( G_1 \) and \( G_2 \) are said to be equivalent.

\[\text{More generally, we can define L-fuzzy languages [63] as an extension of fuzzy languages.}\]
Some Operations on Fuzzy Languages

Let $L_1$ and $L_2$ be two fuzzy languages with membership functions $\mu_{L_1} : \Sigma^* \to [0, 1]$ and $\mu_{L_2} : \Sigma^* \to [0, 1]$ respectively, over an alphabet $\Sigma$, then the following operations hold. Here $\tilde{L}$ be the resulting fuzzy language after each operation with the membership function $\mu_{\tilde{L}}(x)$ [46].

**Union:** The union of $L_1$ and $L_2$ is denoted by $\tilde{L} = L_1 \cup L_2$ and its membership function is defined by $\mu_{\tilde{L}}(x) = \max(\mu_{L_1}(x), \mu_{L_2}(x)), x \in \Sigma^*$.

**Intersection:** The intersection of $L_1$ and $L_2$ is denoted by $\tilde{L} = L_1 \cap L_2$ and its membership function is defined by $\mu_{\tilde{L}}(x) = \min(\mu_{L_1}(x), \mu_{L_2}(x)), x \in \Sigma^*$.

**Complement:** The complement of $L_1$ is denoted by $L_1^C$ or $\tilde{L}_1^\prime$ and its membership function is defined by $\mu_{L_1^C}(x) = (1 - \mu_{L_1}(x)), x \in \Sigma^*$.

**Concatenation:** The concatenation of $L_1$ and $L_2$ is denoted by $\tilde{L} = L_1 \cdot L_2$ and its membership function is defined as follows. Let a string $x$ in $\Sigma^*$ be expressed as a concatenation of a prefix string $u$ and a suffix string $v$, i.e., $x = uv$. Then $\mu_{\tilde{L}}(x) = \max(\min(\mu_{L_1}(u), \mu_{L_2}(v))), u \in \tilde{L}_1, v \in \tilde{L}_2$.

**Kleene closure:** We can readily extend the concept of Kleene Closure to the fuzzy language. Kleene closure of $L_1$ is denoted by $L_1^*$ and is defined by $\tilde{L}_1^* = \varepsilon \cup \tilde{L}_1 \cup \tilde{L}_1 \tilde{L}_1 \cup \ldots$, where $\varepsilon$ is the null string. i.e., $L_1^* = L_1^0 \cup \tilde{L}_1 \cup \tilde{L}_1 \tilde{L}_1 \cup \ldots$.

Note that the meaning of the multiple concatenations $\tilde{L}_1, \tilde{L}_1 \tilde{L}_1, \ldots$ is unambiguous because of the associativity of concatenation.

The $+$ operation on $L_1$ is defined by $\tilde{L}_1^+ = \tilde{L}_1^* - \{\varepsilon\} = \tilde{L}_1 \cup \tilde{L}_1 \tilde{L}_1 \cup \tilde{L}_1 \tilde{L}_1 \tilde{L}_1 \cup \ldots$.

The theory of fuzzy languages offers what appears to be a fertile field for further study. It may prove to be of use in the construction of better models for natural languages and may contribute to a better understanding of the role of fuzzy algorithms and fuzzy automaton in
decision making, learning process of languages, and other processes involving the manipulation of fuzzy data [63]. Also, fuzzy languages can be applied to pattern recognition [64], information retrieval [65].

**Definition 1.8.2.2:** Let \( \tilde{L} \) be a fuzzy language over an alphabet \( \Sigma \) and \( \mu_{\tilde{L}} : \Sigma^* \rightarrow \mathbb{M} \) the membership function of \( \tilde{L} \), where \( \mathbb{M} \) is the set of real numbers in \([0, 1]\). We call \( \tilde{L} \) a regular fuzzy language [66], [67-68] if,

1. the set \( \{ m \in \mathbb{M} \mid S_{\tilde{L}}(m) \neq \emptyset \} \) is finite, and
2. for each \( m \in \mathbb{M} \), the set \( S_{\tilde{L}}(m) = \{ w \mid \Sigma^* \text{ and } \mu_{\tilde{L}}(w) = m \} \) is regular.

Particularly, \( \tilde{L} = \{(w, \mu_{\tilde{L}}(w)) \mid w \in \Sigma^* \} \).

### 1.9 FUZZY AUTOMATA

Since the introduction of fuzzy sets as a method for representing uncertainty, this idea has been applied to a wide range of scientific areas. One such area is automata theory and language theory first introduced by W. G. Wee in [69]. There is a deep reason to study fuzzy automata: several languages are fuzzy by nature (e.g. the language containing words in which many letters “a” occur). The basic idea in the formulation of a fuzzy automaton is that, unlike the classical case, the fuzzy automaton can switch from one state to another one to a certain (truth or membership) degree. Thus, researching fuzzy automaton with ability of processing fuzzy processes is needed. It will process continuous inputs and outputs. Even when a system input at a time is missing, the system can work accurately. Obviously, this system is robust and suitable to pattern recognition, neural networks, lexical analysis, clustering, inference and fuzzy control.

Analogously as in a theory of classical automata there are several definitions of a fuzzy automaton and, hence, there are several categories of these fuzzy automata with possible different properties. Fuzzy automata include deterministic and nondeterministic finite automata as special cases and also have some properties similar to those of probabilistic automata. A fuzzy automaton is a type of automaton which will transit from a state to another state or the same state via the branch whose membership function is the largest one among those of all branches diverging from the state when an input is applied. It can recognize a
fuzzy regular language. An advantage of employing a fuzzy automaton as a learning model is its simplicity in design and computation. The literature on fuzzy automata is given in [70-88].

Fuzzy finite automata have many important applications such as in neural networks [89], automatic control systems [90], pattern recognition [91], for optimizing control of multimodal systems [92], learning systems [69]. Some more applications of fuzzy automata are given in [63] and [79]. Fuzzy automata have been used to deal with fuzzy discrete event systems. These are applied in full-sized signal classification. Another application of fuzzy automaton is some handwritten words recognition. E. S. Santos showed that the capability of a fuzzy automaton as an acceptor is equal to that of finite automaton [73]. The formal definition of fuzzy automaton is given below.

**Definition 1.9.1:** A fuzzy finite automata (FFA) [66] is a 5-tuple \( A = (Q, \Sigma, \mu, S, F) \), where

1. \( Q \) is a finite nonempty set of states,
2. \( \Sigma \) is a finite nonempty set of input symbols,
3. \( \mu : Q \times \Sigma \times Q \rightarrow [0, 1] \) is a function, called the transition function,
4. \( S \) is the initial state,
5. \( F \subseteq Q \) is the set of final states.

\( \Sigma^* \) is the set of all words on \( \Sigma \) and \( \varepsilon \) denotes the empty word in \( \Sigma^* \).

The transition function \( \mu \) can be extended to \( \mu^* : Q \times \Sigma^* \times Q \rightarrow [0, 1] \) by

\[
\mu^*(q, \varepsilon, q) = 1, \mu^*(q, u\varepsilon, s) = \mu^*(\mu^*(q, u, r), a, s) \in [0, 1] \quad \forall q, s \in Q, \forall u \in \Sigma^*, \forall a \in \Sigma.
\]

It can be easily verified that \( \mu^*(q, uv) = \mu^*(\mu^*(q, u, r), v, s) \forall u, v \in \Sigma^* \).

Especially, we call a fuzzy automaton with the grade of transition under the operation “max min” a pessimistic fuzzy automaton (pfa), and a fuzzy automaton under the operation “min max” an optimistic fuzzy automaton (ofa).

**Definition 1.9.2:** Two states of a fuzzy automaton are equivalent if they are

- equivalent in the ordinary sense, i.e., for all input sequences applied to the two states outputs are identical, and
• the grades of acceptance to all input sequences starting from the two states are identical.

**Definition 1.9.3:** Two fuzzy automata $M_a$ and $M_b$ are *equivalent* if for every state of $M_a$ there is at least one equivalent state in $M_b$ and vice versa.

### 1.9.1 MINIMIZATION OF FUZZY FINITE AUTOMATA

Fuzzy finite automata are used to design complex systems. Finding a minimum representation of fuzzy finite automata is critical in such design. The idea of minimizing fuzzy finite automaton was exploited in the papers by Peeva [93], Malik et. al [94], and in Lee [95]. In above papers, fuzzy automaton was reduced by computing and merging indistinguishable states. However, the term minimization used in the mentioned papers does not mean the usual construction of the minimal one in the set of all fuzzy automata recognizing a given fuzzy language. But it is only the procedure of computing and merging indistinguishable states that may not result in a minimal fuzzy automaton.

Fuzzy finite automaton is used to design complex systems. Finding a minimum representation of fuzzy finite automata is critical in such design. State reduction of fuzzy automata has been studied by many authors. All of them have dealt with classical fuzzy automata and reduction has been done using crisp equivalence relations.

Reduction of number of states of fuzzy automata was studied in by means of crisp equivalences, and the algorithms given there were also based on the idea of computing and merging indistinguishable states. They were called minimization algorithms, but the term minimization is not adequate because it does not mean the usual construction of the minimal one in the set of all fuzzy automata recognizing a given fuzzy language, but just the procedure of computing and merging indistinguishable states. Literature on minimizing fuzzy finite automaton is given in [96-101].

In contrast to the deterministic case, where we can effectively detect and merge indistinguishable states, in the non-deterministic case we have sets of states and it seems very difficult to decide whether two states are distinguishable or not [102-104]. There can always be states which could be merged but detecting those is too expensive. In the case of fuzzy automata this problem is even worse because we work with fuzzy sets of states. Myhill-
Nerode theorem is extended for fuzzy automaton [66] and [99] accepting fuzzy regular language and minimization of fuzzy automaton was achieved by the Myhill-Nerode theorem.

1.10 APPROXIMATE STRING MATCHING

Approximate string matching [105] is a recurrent problem in many branches of computer science, with applications to text searching, computational biology, pattern recognition, signal processing, etc. It is an offspring of the much simpler exact string matching problem. Use of the term approximate merely emphasizes the fact that a perfect match may not be achievable and imperfections such as missing and extraneous symbols have to be considered between two strings to be compared. Many fuzzy automaton models have been introduced in the past for imperfect string matching [106-112]. This thesis proposes extended fuzzy automaton models, such as intuitionistic fuzzy automaton model, interval-valued fuzzy automaton model, and vague finite automaton model for approximate string matching using the notion of intuitionistic fuzzy sets, interval-valued fuzzy sets, and vague sets respectively. An algorithm has been proposed for each of the case. The returned value is a percentage that tells how close and far the strings that have been compared. We can convert one model into other and thereby generalize the fuzzy automaton process of approximate string matching. The proposed extended fuzzy automaton methods may be applied in text searching, computational biology, pattern recognition, signal processing, etc.

Approximate string matching is a recurrent problem in many branches of computer science, with applications to text searching, computational biology, pattern recognition, signal processing, etc. The problem of approximate string matching is an offspring of the much simpler exact string matching problem. The problem of exact string matching has been extensively researched. However, approximate string matching is a much more complicated problem to solve and has many more real world applications. Here the general goal is to perform string matching of a pattern string in a text where one or both of them have suffered some kind of (undesirable) change. Instead of searching for the string exactly, approximate string matching searches for patterns that are close to the pattern string. One of the best studied particular cases of approximate string matching is so-called edit distance, which allows to delete, insert and replace simple characters (by a different one) in observed string with respect to the pattern string.
One of the earliest applications of approximate string matching was in text searching. The approximate string matching algorithms can be applied to account for errors in typing. Another application of approximate string matching is in biology. As with text, ideally, exact string matching should be effective. But in reality, DNA searching is not an exact science. There are frequently mutations in DNA that a string matching algorithm must account for. One of the recent applications of string matching is signal processing. This is a problem in which exact string matching can almost never be applied because signals are so easily and often corrupted.

Many classical publications proposing methods for the classification of strings containing errors are available in the literature. Many of them are based on statistical methods. There also exist proposals of fuzzy methods based on fuzzy extensions of the classical theory of formal languages and automata. A fuzzy system can be more intuitive for the evaluation of similarity between two strings, while it needs less strong normalization requirements. Fuzzy modelling efforts are assuming a membership function, which could be regarded as a point estimate of the degree of belongingness, for the reflection of uncertainty of observed string.

1.11 STRUCTURE OF THE THESIS

The thesis is organized chapter wise as follows:

**Chapter 1:** This chapter is introductory and sets up the background for the problems taken up in the thesis. It overviews finite automata, types of finite automata and relation between them, regular language, and minimization of finite automata by stating Myhill-Nerode theorem. The concept of fuzzy sets has been outlined with some basic operations. Three extensions of fuzzy sets such as intuitionistic fuzzy set, interval-valued fuzzy set, and vague set with some of their basic operations including a relation between their membership values are given. Fuzzy grammar and fuzzy language with some of their algebraic operations are also discussed. The concept of fuzzy automaton, the model accepting fuzzy language and its minimization is briefed. Chapter ends with the discussion of approximate string matching in the framework of extended fuzzy automaton.

**Chapter 2:** This chapter proposes intuitionistic fuzzy regular language, an extension of fuzzy regular language using the concept of intuitionistic fuzzy sets. Basic operations such as union, intersection, complement, concatenation, and star operations on these constructed
languages are given. Finite automata (deterministic and nondeterministic) with intuitionistic fuzzy transitions and intuitionistic fuzzy (final) states have been constructed to recognize proposed languages. It is observed that, the finite automata (deterministic and nondeterministic) with intuitionistic fuzzy (final) states are more suitable to recognize intuitionistic fuzzy regular language than the finite automata (deterministic and nondeterministic) with intuitionistic fuzzy transitions. An attempt has been made to express this language through intuitionistic fuzzy regular expressions. Myhill-Nerode theorem is studied in the framework of intuitionistic fuzzy regular language and an algorithm is given to minimize the redundant states of finite automata with intuitionistic fuzzy (final) states.

**Chapter 3:** This chapter introduces interval-valued fuzzy regular language, another extension of fuzzy regular language using the notion of interval-valued fuzzy sets. Some of the basic algebraic operations namely union, intersection, complement, concatenation, and star operations of the introduced language are given. Finite automata (deterministic and nondeterministic) with interval-valued fuzzy transitions and interval-valued fuzzy (final) states automata are proposed to discuss the recognition of interval-valued fuzzy regular language through some theorems. The description of interval-valued fuzzy regular language is attained through interval-valued fuzzy regular expression. Finally, minimization of deterministic finite automata with interval-valued fuzzy (final) states is achieved by proving Myhill-Nerode theorem for interval-valued fuzzy regular language and an algorithm is given for the same.

**Chapter 4:** Vague regular language is proposed in this chapter using the concept vague sets, one more generalization of fuzzy sets. Some algebraic operations on these languages are given. Chapter also proposes finite automata (deterministic and nondeterministic) with vague transitions and finite automata (deterministic and nondeterministic) with vague (final) states to study vague regular language through some theorems. It gives vague regular expressions for useful representation of strings of vague regular language in an algebraic fashion. Furthermore, extended Myhill-Nerode theorem is discussed for vague regular language and an algorithm to minimize deterministic finite automata with vague (final) states is given. While concluding the chapter, a relation between the membership values of the above proposed extended fuzzy languages (in sequence; intuitionistic fuzzy language, interval-valued fuzzy language, and vague language) has been given.
Chapter 5: This chapter discusses the application of proposed extended fuzzy automata in chapter 2, chapter 3 and chapter 4. It presents the approximate string matching using intuitionistic fuzzy automata, interval-valued fuzzy automata, and vague finite automata accepting respectively, intuitionistic fuzzy regular language, interval-valued fuzzy regular language, and vague regular language. These methods, models the possible edit operations such as insertion, substitution and deletion needed to transform an observed string (input string) into a pattern string (target string). The selection of appropriate extension of fuzzy automata models with their membership values for the transitions leads to improve the system performance for a particular application. A relation between the membership values between the pair of strings of extended fuzzy automata is obtained. Chapter ends by giving an algorithm for approximate string matching in each of the three cases.