Formalization for Natural Language Fuzzy Queries and Crisp Multi-Criteria Queries

Waheed Aslam Ghumman
Technische Universität Dresden
Fakultät Informatik
Nöthnitzer Str. 46, 01187 Dresden
Germany
Waheed-Aslam.Ghumman@mailbox.tu-dresden.de

Susana Munoz-Hernandez
Universidad Politécnica de Madrid,
Facultad de Informática
Campus de Montegancedo, 28660, Madrid
Spain
susana@ifi.upm.es

Abstract: It is common in real life to find fuzzy information that comes from subjective judgments or the imprecision in measured data. Fuzzy approaches have been used to extend database systems in storing and updating imprecise information (data) and in processing imprecise queries. Consider a fuzzy query: find name, grade of quite good students and just tall students where age > 15. This query includes two fuzzy concepts: good student and tall student and one crisp query criteria (i.e. age > 15). In this paper we present a formalization to process natural language fuzzy (expressive) queries and to return fuzzy results for crisp query criteria. Our formalization is general that can be particularized for implementation in variety of database platforms i.e. fuzzy web search, information systems supporting fuzzy data etc. Our approach only makes the fuzzy query writing much simpler and easier than conventional query writing but also close to human like thinking due to its true fuzzy nature. We also provide an operational semantics for fuzzy query processing which can be followed for multiple data types i.e. numeric, text, graphics etc. Our approach supports fuzzy querying for not only fuzzy data but also for missing data; hence enabling us to get query results closer to human thinking and expectations. It is an expressive model that let to make human-like (i.e. fuzzy) consults.

Key–Words: Fuzzy Databases, Fuzzy Queries, Fuzzy Logic

1 INTRODUCTION

Consider a database table students with attributes(columns) name, age, height and grade. Suppose we query students table with the fuzzy query: find name, grade of quite good students and just tall students where age > 15. This query includes two fuzzy concepts i.e. good student and tall student. Additionally this query includes a crisp search criteria (i.e. age > 15). In this paper, we formalize a database model which can process natural language fuzzy queries (i.e. good student, tall student) and also will return such results for crisp search criteria which not only fulfill crisp criteria completely but also such results which fulfill crisp search criteria partially. We introduce a new concept of fuzzy modifiers(e.g. quite, just, very) along with fuzzy concepts (e.g. tall, good, young), which enable processing natural language fuzzy queries. Such kind of fuzzy model can be used to build information retrieval systems, document retrieval systems and web searching systems supporting fuzzy approach. Much of the human reasoning is based on fuzzy reasoning. In fact, fuzzy data arise constantly in real life from human thought and cognition processes and we often make decision based on them [1]. However, classical information systems handle only crisp, precise and non-ambiguous data. Fuzzy querying can be defined as; a computer search that returns not only exact matches to the search request, but also close matches that include possibilities. In most of the cases the word possibilities in definition is interpreted as possibilities of spelling mistakes and/or to search for close matches with respect to synonyms but in real world fuzzy concepts are more complex than spelling mistakes and synonyms e.g. while querying a students database, we might be interested to search for tall students to select for college basket ball team. In this case, if we query our database by giving just crisp student height limit (i.e. height $\geq$ 190cm) then there might be a chance that there are not enough students who has height $\geq$ 190cm, so we may not get enough students for basket ball team. On the other hand, by using a fuzzy query we can get not only such students who have height $\geq$ 190cm but also such students who are close to that height i.e. 180cm or 185cm.
1.1 Existing Approaches
Various models and prototypes have been proposed which make use of fuzzy set theory [2] and handle fuzzy data in different ways. Bookstein [3] proposes relative importance of various index terms in a boolean request while retaining the desirable properties of a Boolean system. Bookstein uses the concept of fuzzy sets to form a fuzzy request in which weights are assigned to index terms in the range of [0, 1]. Nomoto et al. [4] develop a fuzzy document retrieval system using fuzzy graph theory in which citations of documents are chosen as the criterion for fuzziness. Citation network, i.e., graph comprises citations and relation between them is created for fuzzy retrieval. In [5], [6], and [7], these authors introduce the fuzzy extension of several major EER concepts (super class, subclass, generalization, specialization, category, and shared subclass) without including graphical representations. They discuss three kinds of constraints with respect to fuzzy relationships but they do not study fuzzy constraints. Ribeiro et al [8] presented a flexible query interface (based on fuzzy logic) where queries in natural language with pre-defined syntactical structures are performed, and the system uses a fuzzy natural language process to provide answers. We also include support for natural language based fuzzy queries. Many others have presented/developed different fuzzy databases for document retrieval and information retrieval systems [9], [10] and [11].

In this paper, we present a fuzzy querying based database model which is different from earlier ones in multiple ways i.e. we introduce some new concepts like fuzzy modifiers and our model combines many aspects of a true nature of fuzzy querying i.e. formalism for processing natural language based queries and returning fuzzy results for crisp criteria in query etc. This model also follows an abstract and generic approach which can be implemented for variety of information retrieval systems and document retrieval system dealing with fuzzy data/information. For example, Nilufar et al.[12] present a content-based visual image retrieval system that can retrieve an image using a fuzzy description of colors and textures, the method presented by Nilufar et al.[12] can be used in our model for defining the fuzzy functions and for querying an images database in a fuzzy way.

1.2 Fuzzy Concepts
There are many fuzzy concepts in real life such as being young, cheap, tall or rich that are relative and cannot be represented in a crisp way. We can relate individual with this fuzzy concept using a truth value (from the interval [0,1]) and determine that John is tall with a truth value of 0.8 (i.e. John is quite tall), but Sara is tall with a truth value 0.5 (because she is just tall). In this example, tall is a fuzzy concept and keywords [quite and just] are named as fuzzy modifiers. These fuzzy concepts can be defined using fuzzy functions or generally called membership functions. Usually, fuzzy functions provide a truth value associated to a domain value that let us to calculate the value for the fuzzy concept. For example, we can approximate the truth value of young from the age (see figure 1), we can approximate the truth value of being tall from the length etc. We are going to call these measurable parameters (age, length, speed etc) as attributes that let us define a mathematical function called fuzzy function to describe a fuzzy concept (young, tall, fast, good student etc). So each fuzzy concept is mapped on some (measurable parameter) attribute.

As we have mentioned in introduction that our database model will focus on two aspects of fuzzy query processing i.e. one is to process the natural language fuzzy criteria in fuzzy query (e.g. quite good student) and other aspect is to process crisp criteria in fuzzy query to return fuzzy results. So, in section 2, we will present formalization for natural language fuzzy queries and in section 3, we will present formalization about returning fuzzy results for crisp query criteria. Section 4 presents a brief overview, processing flow of our model, interpretation of fuzzy query from natural language to system understandable format (i.e. logical fuzzy query) and also detailed processing for fuzzy queries in form of a fuzzy query processing algorithm (which also combines the two formalizations presented in section 2 and 3). In section 5, we will give brief analysis of our approach and we will discuss how our fuzzy database model can be easily implemented for multiple kinds of databases dealing with various types of contents. In section 6, we will give conclusions and future work.

2 Formalization for Natural Language Fuzzy Criteria
We will give some preliminary definitions and concepts first which are necessary in final formalization. In rest of the paper the word attribute represents a column/field of a database table i.e. students table may have attributes: name, age, grade, height etc.

Definition 1 Fuzzy Function or Membership Function
A function $\delta$ is a fuzzy function (also termed as membership function) if $\delta(X_o) = V$, where $X_o$ belongs to the domain of a measurable attribute or it is conjunction of measurable attributes and
Figure 1: Truth value graph of fuzzy concept young

\[ V \in [0, 1]. \] Semantically, \( V \) represents the truth value of \( X_o \) with respect to fuzzy concept that defines \( \delta \).

Definition 2 Return Attributes List

Return attributes list \([RA_1, RA_2, ..., RA_n]\) is the list of such attributes (of database table \( T \)) which are required in result set of a query. We will use abbreviation \([RA_s]\) to represent the list of return attributes and \( RA_i \in [RA_s] \) represents the \( i \)-th element of \([RA_s]\).

Example 1

For example, we have a query: find name, grade of students where age > 15. In this example only name and grade are elements of return attributes list, as query requires that name and grade of students should be returned, attribute age is used just as search criteria.

2.1 Fuzzy Pairs

As we mentioned in introduction that fuzzy modifiers are natural language words which determine the intensity of a fuzzy concept to which they are attached. Examples of fuzzy modifiers can be words like very, quite, just etc i.e. Suppose we attach fuzzy modifier quite and just to a fuzzy concept tall, we get quite tall and just tall (both resulting in different interpretations). For simplicity, we name this pair (fuzzy modifier + fuzzy concept) as fuzzy pair denoted by \( FP \). As one query may include \( n \) \((n \geq 0)\) number of fuzzy pairs, so we represent list of fuzzy pairs by \([FP_s]\) and \( FP_i \in [FP_s] \) represents \( i \)-th element of \([FP_s]\).

Implementation of Fuzzy Pairs:

Each fuzzy pair is defined in database as a separate table \( (FPT) \) and each \( i \)-th row \((i \geq 1)\) of \( FPT \) has four data values i.e. \( X_{i1}, X_{i2}, V_{i1}, V_{i2} \), such that:

- \( X_{i1} \) and \( X_{i2} \) represent a continuous interval \([X_{i1}, X_{i2}]\).
- \( V_{i1} \) and \( V_{i2} \) represent a continuous interval \([V_{i1}, V_{i2}]\).

Interval \([V_{i1}, V_{i2}]\) represents range of truth values associated with the range of domain values \([X_{i1}, X_{i2}]\). Following example explains this concept.

Example 2

For example we have a fuzzy pair quite young (i.e. fuzzy modifier quite and fuzzy concept young which is mapped on age attribute of table student). Suppose we say that:

- A person is quite young with a truth value 0 if person’s age is less than or equal to 0.
- Truth value for a person being quite young increases from 0 to 1 with age increasing from 5 to 15 i.e. according to this rule person is 50% young if his/her age is 10 years.
- Person is quite young with truth value 1 if person has age between interval [15,25].
- Truth value for a person being quite young decreases from 1 to 0 with age increasing from 25 to 35 i.e. according to this rule person is 50% young if his/her age is 30 years.
- A person is quite young with a truth value of 0 if person’s age is 35 or more.

Then we can define some intervals in form of a table in database for quite young as shown in table 1. First two columns (Age1 and Age2) represent two bounds of a continuous interval of age attribute i.e. data values 5 and 15 in second row represent an interval [5,15] (symbolically represented as \([X_{21}, X_{22}]\)), next two columns (TruthValue1 and TruthValue2) represent the two bounds of continuous interval of...
So, data in second row (i.e. 5, 15, 0, 1) of table 1 means that quite young fuzzy pair has truth value of 0 at age of 5 years and quite young has truth value of 1 at age of 15 years. Each fuzzy pair table will have four values $X_{i1}$, $X_{i2}$, $V_{i1}$, $V_{i2}$ in each row to determine truth value for any input value $X_o$ e.g. if we want to know that what is truth value of the statement: A person with age 30 years is quite young, then as 30 (we name this input value as $X_o$) falls in interval $[25,35]$ which has corresponding truth value interval $[0,1]$ (see table 1), so according to this information, truth value for this statement will be 0.5. Formal method for exact computation of truth value for any given input parameter $X_o$ based on $[X_{i1}, X_{i2}]$ and $[V_{i1}, V_{i2}]$ is given in section 2.2. Different fuzzy pair tables can be named using a standard naming convention, so that (in runtime environment) system can decide automatically that which fuzzy pair table to access based on the fuzzy modifier and fuzzy concept used in the query. So, fuzzy pair tables can be named using following rules:

1. Table name starts with $FPT_-$ (abbreviated from fuzzy pair table).

2. Append table name with identifiers fuzzy modifier name and fuzzy concept name (i.e. $FPT_{FuzzyModifierFuzzyConcept}$).

As each fuzzy concept is mapped to some measurable attribute of related table, so we will also define naming convention to store this mapping in a mapping table in database which will also be helpful to generate the name for this table automatically in runtime. The records of this table will be: fuzzy concept in first column and the measurable attribute (to which this fuzzy concept is mapped) in second column. So, we will name this mapping table as following:

- Table name starts with $FCM_-$ (abbreviated from fuzzy concept mapping).

- Suppose, fuzzy concept is mapped to an attribute $A$ of $TABLE_X$ then mapping table name will be as $FCM_{TABLE_X}$.

2.2 Extended Fuzzy Function

As discussed in section 2.1 that we need four parameters $X_{i1}$, $X_{i2}$, $V_{i1}$ and $V_{i2}$ to decide the truth value of input parameter $X_o$. So, we extend the basic fuzzy function defined in definition 1. Before giving the formula for fuzzy function $\delta_f(X_o, X_{i1}, X_{i2}, V_{i1}, V_{i2}, V_f)$, which computes the truth value $V_f$ for input parameter $X_o$, we will give some necessary definitions and notations to be used for this purpose.

Although, there are multiple types of data, we are going to focus on the numeric data only because the definition of fuzzy function for the rest of the types is done by the enumeration of the truth values for each individual of the domain. Nevertheless, the process to model a fuzzy function for a numerical domain is quite general and we are formalizing with following definitions.

**Definition 3 Success Interval**

We say that $[X_{i1}, X_{i2}]$ is a success interval of a fuzzy function $\delta_f$, if $V_{i1} = V_{i2} = 1$.

**Definition 4 Failure Interval**

We say that $[X_{i1}, X_{i2}]$ is a failure interval of a fuzzy function $\delta_f$, if $V_{i1} = V_{i2} = 0$.

**Definition 5 Increasing Interval**

We say that $[X_{i1}, X_{i2}]$ is an increasing interval of a fuzzy function $\delta_f$, if: $X_{i1} < X_{i2}$ and $V_{i1} < V_{i2}$ and $\delta_f$ is continuous.

**Definition 6 Decreasing Interval**

We say that $[X_{i1}, X_{i2}]$ is a decreasing interval of a fuzzy function $\delta_f$, if: $X_{i1} < X_{i2}$ and $V_{i1} > V_{i2}$ and $\delta_f$ is continuous.

Using above definitions, extended fuzzy function $\delta_f$ for numeric type attributes is given in equation 1.

**Definition 7 Formal Fuzzy Query**

Formal fuzzy query is a hybrid query containing natural language fuzzy pairs (defined in section 2.1) and/or crisp criteria (either one of them or both). Formal fuzzy query has following components:

1. Return attribute(s)

2. Fuzzy Pair(s)

3. Table name

4. Crisp criteria.

**Example 3**

An example of formal fuzzy query can be: Find name, age, grade of very young and quite tall students where grade==B. In this example, [name, age, grade] are return attributes, [very, quite] are fuzzy modifiers, [young, tall] are fuzzy concepts, ’students’ is table name and grade == B is crisp criteria.
\[ \delta_f(X_0, X_{i1}, X_{i2}, V_{i1}, V_{i2}, V_f) = \begin{cases} 
V_f = 1 & \text{if } X_0 \in [X_{i1}, X_{i2}] \\
& \text{and } [X_{i1}, X_{i2}] \text{ is success interval} \\
V_f = \frac{X_0 - X_{i1}}{X_{i2} - X_{i1}} & \text{if } X_0 \in [X_{i1}, X_{i2}] \\
& \text{and } [X_{i1}, X_{i2}] \text{ is an increasing interval} \\
V_f = \frac{X_{i2} - X_0}{X_{i2} - X_{i1}} & \text{if } X_0 \in [X_{i1}, X_{i2}] \\
& \text{and } [X_{i1}, X_{i2}] \text{ is decreasing interval} \\
V_f = 0 & \text{if } X_0 \in [X_{i1}, X_{i2}] \\
& \text{and } [X_{i1}, X_{i2}] \text{ is failure interval} \\
V_f = 0 & \text{Otherwise} 
\end{cases} \]

Figure 3: Extended fuzzy function
Definition 8 Logical Fuzzy Query

If:

- \([RA_s]\) represents list of return attributes (see definition 2).
- \([(\text{attribute, operator, value})]\) represents one element from the list of crisp criteria \([CC_s]\), operator can be any comparison operator \((<, >, \leq, \geq, =, \neq, \text{etc.})\), logical operator ("like", "IN" etc.) or any standard query operator.
- \(FP_s\) represents list of fuzzy pairs \([FP_s]\) (see section 2.1).

Then logical fuzzy query is a well defined syntactical structure LFQ as following:

\[ LFQ([RA_s], [FP_s], \langle \text{table name} \rangle, [CC_s]). \]

Continuing with the previous example 3 in above definition, corresponding logical fuzzy query will be:

\[ LFQ(\langle \text{name, age, grade} \rangle, [(\text{very, young}), (\text{quite, tall})], \text{students}, [(\text{grade, =, B})]). \]

3 Fuzzy Formalization for Crisp Query Criteria

Suppose, we have a table \(T\) with an attribute \(A\) containing some data values \([DV_s]\) and we have a query which includes a crisp search criteria as \((A(\text{comparison operator}) X_c)\), then each data value \(X_i \in [DV_s]\) will be compared with given crisp criteria to decide whether \(X_i\) fulfills search criteria or not. Generally, crisp search returns results only if criteria is fulfilled completely but we will define crisp fuzzy function \(\alpha_c(X_i, X_c, \text{Operator}, DF, DP, V_c)\) which will ensure such results that fulfill search criteria partially (\(\text{Operator}\) in function parameters represents the comparison operator used in crisp criteria). First five parameters of \(\alpha_c\) are input parameters and \(V_c\) (output parameter) is truth value (showing the partial or complete match). \(V_c = 1\) for complete match of data value \(X_i\) with given crisp criteria. In database, we will store two numbers as difference factor \(DF\) and decrease percentage \(DP\) where \(DP \in [0, 1]\) for attribute \(A\) such that truth value \(V_c\) will be decreased by \(DP\) for every unit difference \(DF\) between \(X_c\) and \(X_i\). We can store \(DF\) and \(DP\) in a table in database with naming convention as \(CCF_T\) (abbreviated from crisp criteria factors) where \(T\) is table name containing attribute \(A\) which is to be included in crisp criteria. So each row of \(CCF_T\) will be a tuple containing attribute name \(A, DF\) and \(DP\). Each attribute \(A \in T\) which can be included in crisp criteria should be part of \(CCF_T\).

So crisp fuzzy function \(\alpha_c\) to calculate \(V_c\) for the given input parameters \(X_i, X_c, \text{Operator}, DF, DP\), is given in equation 2.

Example 4

For example, we have a table \(\text{students}\) in database with attributes \(\text{name, age}\) and data as shown in table 2 and we query our database with a simple query: find name of students where age > 15 (in this query \(X_c = 15\)). As we start reading data from \(\text{students}\) table, and we read value \(X_i\) from \(\text{age}\) attribute of student table, we get \(X_i = 4\) as first student has \(age = 4\), which does not fulfill or query criteria (i.e. \(age > 15\)). Similarly second row will also not fulfill the query criteria and we will get only one record in our result set as \(\text{Alice}\) has age of 16 years.

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>12</td>
</tr>
<tr>
<td>Alice</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 2: Students data

Now we will take two parameters (i.e. decrease factor \(DF\) and decrease percentage \(DP\)) as \(DF = 1\) and \(DP = 0.10\) (i.e. \(10\%\)), which represent that with each unit difference \(DF\) of 1 year between the \(X_i\) and \(X_c\), truth value \(V_c\) will be decrease by \(10\%\) (as \(DP\) is \(0.1\)). So to get fuzzy results, we will use our crisp fuzzy function \(\alpha_c\) for each row as following:

For row 1: we have \(\alpha_c(X_i = 4, X_c = 15, >, DF = 1, DP = 0.1, V_c)\)
So, \(V_c = 0\). (Please use definition of \(\alpha\) given in equation 2 to calculate \(V_c\)).

For row 2: we have \(\alpha_c(X_i = 12, X_c = 15, >, DF = 1, DP = 0.1, V_c)\)
So, \(V_c = 1 - \frac{\|2-15\times0.1\|}{1} = 0.7\)

For row 3: we have \(\alpha_c(X_i = 16, X_c = 15, >, DF = 1, DP = 0.1, V_c)\)
So, \(V_c = 1\)

So, according to our fuzzy search we will get two rows in which we get \(V_c > 0\), one row fulfills our crisp criteria completely having \(V_c = 1\). As second row is close to the crisp criteria given in query, so according to our defined rules \((DF\) and \(DP\)) it returns truth value \(V_c\) of 0.7.

4 Fuzzy Database Model

Now that we have most of the concepts in place, we are ready to present an overview of our database.
\[ \alpha_c(X_i, X_c, \text{Operator}, DF, DP, V_c) \]

\[
\begin{align*}
V_c &= 1 & \text{if } X_i < \text{Operator} > X_c \text{ is true.} \\
V_c &= 1 - \frac{|X_i - X_c| \times DP}{DF} & \text{if } X_i < \text{Operator} > X_c \text{ is false, AND } (V_c = 1 - \frac{|X_i - X_c| \times DP}{DF}) > 0. (2) \\
V_c &= 0 & \text{Otherwise.}
\end{align*}
\]

Figure 4: Crisp fuzzy function

model and detailed fuzzy query processing. We have shown brief overview of fuzzy query processing in figure 5. We will receive a formal fuzzy query in start which will be transformed into a logical fuzzy query by query interpreter (section 4.1). Then fuzzy query processor will process logical fuzzy query to return the result set. Fuzzy query processor reads data from relevant database table which will be processed by extended fuzzy function and crisp fuzzy function to return a combined truth value \( V \). Following sections explain this processing in detail.

### 4.1 Query Interpreter

Query interpreter works as basic interaction platform between user and the rest of database model. Input to query interpreter is a formal fuzzy query (definition 7) through some query interface or query file (depending on the database system being used) and output is a logical fuzzy query (definition 8). So, query interpreter transforms the formal fuzzy query into logical fuzzy query. Transformation process can be defined by a trivial algorithm which reads the formal fuzzy query and transforms it into a logical fuzzy query using some keyword tokens to recognize different components of the formal fuzzy query. Miroslav [13] presented a fuzzy query interpreter which transforms fuzzy queries to the classical SQL structure and queries based on linguistic expressions on client side are supported. Many others (e.g. [14], [15]) present fuzzy query interpreter in their approaches towards fuzzy query processing.

### 4.2 Fuzzy Query Processor

Fuzzy query processor takes a logical fuzzy query as input (converted by query interpreter as discussed in section 4.1) and processes it to return a result set \( RS \) as output. Logical fuzzy query has following structure as discussed in definition 8:

\[ \text{LFQ}(\langle RA_s \rangle, \langle FP_s \rangle, \langle \text{table name} \rangle, \langle CC_s \rangle) \]

So, our goal is to process the logical fuzzy query and return a result set \( RS \). Result set \( RS \) will contain data for all return attributes of successfully selected rows and also a truth value \( V \) for each such row. Main approach for fuzzy query processing is given below:

1. We read each row \( r \) (one by one) of the table \( T \) mentioned in \( \text{table name} \) parameter of \( \text{LFQ} \).

2. For each row \( r \), we will find the truth value \( V \) which determines whether row \( r \) matches with our natural language fuzzy criteria and crisp criteria or not. We will process natural language fuzzy criteria using our extended fuzzy function \( \delta_f \) and crisp criteria will be processed by crisp fuzzy function \( \alpha_c \). Both functions \( \delta_f \) and \( \alpha_c \) will return truth values \( V_f \) and \( V_c \) respectively.

3. As, we can have more than one fuzzy pairs as search criteria in one query, so row \( r \) will be examined against each fuzzy pair such as:

   - Take one element \( FP \) from fuzzy pair list \( FP_a \), one element of fuzzy pair list is a tuple (fuzzy modifier, fuzzy concept).
   - As each fuzzy concepts is mapped to a specific attribute (column) \( A \) of the table \( T \), so we will take value \( X_o \) of row \( r \) and attribute \( A \).
   - Generate name of fuzzy pair table \( FPT \) based on the information in tuple (fuzzy modifier, fuzzy concept) and following the naming rules mentioned in section 2.1.
   - Get the values \( X_{i1}, X_{i2}, V_{i1} \) and \( V_{i2} \) from the \( FPT \). Selection of values \( X_{i1}, X_{i2}, V_{i1} \) and \( V_{i2} \) from fuzzy modifier table will depend on the rule: \( X_{i1} \leq X_o \leq X_{i2} \).
   - These values \( (X_{i1}, X_{i2}, V_{i1} \text{ and } V_{i2}, X_o) \) will be passed to \( \delta_f \) which will determine
Figure 5: Overview of fuzzy query processing in our model
the truth value for the given data value $X_o$. As there can be more than one fuzzy pairs in $[FP_s]$, so for each fuzzy pair one truth value $V_{f_i}$ will be determined by $\delta_f$. Suppose we have $n (n \geq 1)$ number of fuzzy pairs in $[FP_s]$ then truth value $V_f$ for each row will be:

$$V_f = \sum_{i=1}^{n} V_{f_i}.$$  

- If $V_f > 0$ then it means, row $r$ matches with our natural language fuzzy criteria.

4. Similarly, we will find $V_c$ for row $r$ based on all elements of crisp criteria list $[CC_c]$.  

5. We will find average values for $V_f$ and $V_c$ which will lead to find an overall truth value $V$ as shown in algorithm in figure 4.2.  

6. If $V > 0$ (which means that at least one natural language fuzzy criteria or crisp criteria returned fuzzy result successfully for row $r$) then we will add data values of return attributes of row $r$ and $V$, to our result set RS.  

7. Repeat the process for each row of table T.

As, each row of the result set also contains truth value truth value $V$, so result set can be sorted in descending order of $V$ column before displaying results to end user. Result with highest value of $V$ is displayed on top (highest value of $V$ means that row with closest match with natural language fuzzy criteria and crisp query criteria will be displayed at top). The complete algorithm for fuzzy query processing is shown in figure 4.2.

5 Analysis and Benefits of our Approach

Most of the current fuzzy database systems are implemented in such a way that values are hard coded in a fuzzy query, but in this paper we introduce a new concept of fuzzy pairs (consisting a fuzzy modifier and fuzzy concept) which enable us to write natural language fuzzy queries at front end (rather than hard coding values). Additionally, our approach helps in handling missing data in some parts of a row i.e. if a row satisfies at least one fuzzy criteria or crisp criteria, then this row will be surely included in result set even data is missing in some attributes.

Example 5

In these days, many websites for airline ticket reservation have in the form of fuzzy concepts e.g. user might be asked to select preferred flight time from a list having options like [Early Morning, Morning, Noon, Afternoon, Evening, Night, Late Night]. Generally this kind of fuzzy concepts are implemented as hard coding the ranges of domain values against each fuzzy concept in query. i.e. implemented query for fuzzy concept Early Morning may look like find flights with flight_time ≥ 0400 and flight_time ≤ 0700.

Maintenance cost of such systems is very high as (with a slight change in criteria of fuzzy concept) we will have to change the source code for each query involving that fuzzy concept. Our approach defines ranges of data values in a separate table in database and queries are written in a pure natural language fuzzy way which makes database maintenance cost minimum. Another important aspect of our approach is that it can process multiple fuzzy concepts and crisp multi criteria queries at the same time. Stephan et al. [16] present a Skyline operator which selects the best rows or all non-dominated based on a crisp multi-criteria comparison. A row dominates another one if it is as good or better than the other in all multiple criteria and better in at least one criterion. In Skyline, multi-criteria must be satisfied simultaneously in order to obtain the best rows. Marlene and Leonid [17] in their paper point out that Skyline has another problem: dominance rigidity. There is no distinction between rows that are dominated by fare and those that are near to dominant rows. Additionally, Skyline answers are not discriminated due to the fact that Skyline is based on crisp comparison. Skyline rigidity could be solved by means of fuzzy logic. So Marlene and Leonid[17] propose a new Fuzzy Dominance Skyline operator in order to solve Skyline rigidity. It relaxes Skyline by means of fuzzy dominance comparisons among rows. Thus, a Fuzzy Dominance Skyline query selects and discriminates rows that are actually not dominated, but also rows that are near to be not dominated. Our approach not only returns fuzzy result for crisp multi-criteria queries but also handles natural language fuzzy concepts. It is difficult to know whole data and to modify crisp criteria each time to get required results, so fuzzy search can be really beneficial.

5.1 Implementation for Complex Database Systems

In above sections, we have presented formalization considering only one table in database and numeric data only. But, this formalization can be easily im-
ResultSet FuzzyQueryProcessor($LFQ([RA_s],[FP_s],Table_T,[CC_s])$)
{
    Result set $RS$.
    For each Row $r$ of Table$T$
    {
        $V = 0$.
        $V_f = 0$.
        $V_c = 0$.
        For each FuzzyPair $FP$ in $[FP_s]$
        {
            Fuzzy Concept $FC = \text{Get fuzzy concept name from } FP.$
            Fuzzy Modifier $FM = \text{Get fuzzy modifier name from } FP.$
            Attribute $A = \text{Read attribute for } FC \text{ from } FCM_{Table_T}.$
            /*$FCM$ denotes fuzzy concept mapping table
            see Fuzzy Pairs section for details.*/
            Data Value $X_0 = \text{Read data value from row } r \text{ of attribute } A.$
            Read values $X_{i1}, X_{i2}, V_{i1}, V_{i2}$ from FPT$_{FM_{FC}}$.
            //see Fuzzy Pairs section for details.
            $V_f += \delta_f(X_0,X_{i1},X_{i2},V_{i1},V_{i2}).$
            /*Call Extended Fuzzy Function (we omit output parameter, as it will be return value of function)*/
        }
        For each CrispCriteria $CC$ in $[CC_s]$
        {
            Read Attribute $A$, $X_c$, Operator from tuple $CC$.
            Read DP and DF from CCF$_{Table_T}$.
            Data Value $X_i = \text{Read data value from row } r \text{ of attribute } A.$
            $V_c += \alpha_c(X_i,X_c,Operator,DF,DP).$
            /*Call Crisp Fuzzy Function (we omit output parameter, as it will be return value of function)*/
        }
        $V_f = V_f / \text{(number of elements in } [FP_s] \text{ list}).$
        $V_c = V_c / \text{(number of elements in } [CC_s] \text{ list}).$
        $V = (V_f + V_c)/2.$
        If ($V > 0$)
        {
            $RS += \text{data values of row } r \text{ from all attributes in } [RA_s],$
            and append truth value $V$ as an additional column.
        }
    }
    Return $RS$.
}

Figure 6: Query processing algorithm
implemented for other data types like text, date etc. For example, if we have more than one database tables to be considered then as a first step, we will get a new table by joining all the tables in queries using standard database join techniques (see [18] for detail) and then we can apply same formalization on that new single table. We can implement this formalization for other data types also. For example, if we have text data then we can generate a numeric enumeration for this textual data which can be used in computing the fuzzy results. Similarly, we can implement this database model for complex data types e.g. graphics, documents etc. For example, if we are dealing with an image database, then we can set some image evaluation parameter which evaluates the image and returns a numeric result. Then this numeric result can be implemented using our model in such a way that fuzzy query for images will evaluate each image to get that numeric result which can be further used to return fuzzy results. Similarly, this database model can be implemented for web based search systems supporting fuzzy searches. For example, we may give input to a web search engine like: find good rugby players in countryX, then a fuzzy search engine may be implemented in such a way that it computes the fuzzy results (for fuzzy concept good player) from information available for rugby players in countryX.

6 Conclusions and Future Work

This model presents a very simple yet very effective approach in fuzzy querying. It can be implemented for a variety of databases, information systems and document retrieval systems. We have provided formalization for arithmetic data based fuzzy querying which processes not only natural language fuzzy criteria but also returns fuzzy results for crisp multi-criteria queries. We have presented formalization for both type of criteria in fuzzy queries. We have presented a unique method to implement fuzzy concepts which makes a fuzzy query close to human thinking and independent of crisp ranges of domain values. Using the same model and same fuzzy querying algorithm, most of the general data types can be implemented by enumerating the members of domain. For more complex data types, graphics, documents etc, same formalization can be implemented. For example, to define the fuzzy membership function for complicated objects (such as images), we can use work of Ronald Fagin in [19] (which discusses the issues about fuzzy queries in multimedia database systems). We are working on multiple dimensions of this model including implementation of this model for a web search engine which supports fuzzy queries for multiple kinds of information available around the globe i.e. to implement a model for web search engine which supports fuzzy queries and computes the fuzzy results based on single generic algorithm, without having fuzzy pair mapping tables and with domain independent rules. Future work also includes to enhance and analyze this model for different comparison operators in crisp multi-criteria queries and also to study the behavior of crisp fuzzy function with respect to different conjunction and disjunction operators.

References:


