Final Report

A Fuzzy XML Database System

Researchers

- Asst. Prof. Apirada Thadadech

  Department of Computer Science, Faculty of Science, Prince of Songkla University
- Asst. Prof. Preecha Vonghirandecha

  Department of Computer Science, Faculty of Science, Prince of Songkla University
- Asst. Prof. Dr. Supaporn Kansomkeat

  Department of Computer Science, Faculty of Science, Prince of Songkla University
- Assoc. Prof. Dr. Srdjan Skrbic

  Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia

โครงการวิจัยนี้ได้รับทุนสนับสนุนจากเงินรายได้มหาวิทยาลัยสงขลานครินทร์

ประจำปีงบประมาณ.....2557.....รหัสโครงการ..... SCI570329S.....

Acknowledgements

We would like to acknowledge the support of the budget revenue from Prince of Songkla University and Faculty of Science, Prince of Songkla University, Thailand, through the project no. SCI570329S: A Fuzzy XML Database System.

We would like to express my thanks to Department of Computer Science, Faculty of Sciences, Prince of Songkla University for giving us places and time to do this research.

We would also like to thank Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad for inviting, working together, sharing knowledge and collaborating.

Abstract

In recent times, XQuery is the standard language for querying XML documents. However, data in the real world are imprecise and vague values, but standard XQuery does not facilitate to support these kinds of data. Therefore, we present an approach which uses fuzzy set theory to manage these data to XML technology. The objective of this paper is to extend XQuery, namely "fuzzy XQuery", for providing *priority*, *threshold* and *fuzzy* expressions. An interpreter for fuzzy XQuery is developed by using GPFCSP concept, Java programming language and eXist-native XML database.

บทคัดย่อ

ปัจจุบันนี้ ภาษาสอบถามเอ็กคิวรี (XQuery) เป็นมาตรฐานของภาษาสอบถามที่ใช้ในการสืบค้นเอกสารเอ็กเอ็มแอล (XML) อย่างไรก็ดี ในความเป็นจริงข้อมูลส่วนใหญ่จะมีลักษณะไม่ชัดเจนและคลุมเครือ แต่ภาษาสอบถามเอ็กคิวรีไม่สนับสนุนการทำงานกับข้อมูลเหล่านี้ ดังนั้น ในงานวิจัยนี้ จะนำเสนอการใช้ทฤษฎีฟัซซีเซตเพื่อจัดการกับข้อมูลเหล่านี้ในเทคโนโลยีเอ็กเอ็มแอล วัตถุประสงค์ของงานคือต้องการจะขยายความสามารถของภาษาสอบถามเอ็กคิวรี ซึ่งให้ชื่อว่า ภาษาสอบถามเอ็กคิวรีแบบคลุมเครือ เพื่อที่ผู้ใช้สามารถระบุเงื่อนไขในการสอบถามโดยใช้นิพจน์ลำดับความสำคัญ (priority expression) นิพจน์ขีดกั้น (threshold expression) และนิพจน์คลุมเครือ (fuzzy expression) ได้ ตัวแปลภาษาที่ใช้ในการประมวลผลภาษาสอบถามเอ็กคิวรีแบบคลุมเครือได้ถูกพัฒนาขึ้นโดยใช้หลักการ GPFCSP ภาษาจาวาและฐานข้อมูล eXist-db

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Most of the real world information comes in the form of imprecise or incomplete values. Tools for fuzzy logic usage with relational databases have been studied deeply in the past, but in recent years, fuzzy database community interest has been shifted to usage of fuzzy logic with XML. Although several directions of research have been undertaken, there is a lack of stable, usable implementations and standardized fuzzy extensions. As a consequence, there is a great interest in developing standardized, industry usable extensions to XML databases and XML languages using fuzzy logic.

Furthermore, the concept of constraint satisfaction problem (CSP) has been known for years. The aim of CSP is to find a solution that satisfies all the constraints in optimal time. If the satisfaction of the constraint is not a Boolean value, i.e., if there can be many levels of constraint satisfaction, it is clear that there is room for inserting fuzzy values and fuzzy logic into CSP. We can model constraints as fuzzy sets over a particular domain. This leads to fuzzy constraint satisfaction problem (FCSP) reference. Obviously, the degree of satisfaction of a constraint is the membership degree of its domain value on the fuzzy set that represents it. In order to obtain the global satisfaction degree, we need to aggregate the values of each constraint. For the aggregation operator we can use operators from fuzzy logic: t-norms, t-conorms and strict negation. Priority is generally viewed as the importance level of an object among others and it is often used in real time systems. PFCSP is actually a fuzzy constraint satisfaction problem (FCSP) in which the notion of priority is introduced. After that the PFCSP was generalized to GPFCSP (Generalized Prioritized Fuzzy Constraint Satisfaction Problem). The definition of GPFCSP is the same as the definition of PFCSP. However, the GPFCSP adds the possibility to use a disjunction and negation.

The eXtensible Markup Language (XML) is becoming the standard for data description and exchange between various systems and databases over the Internet. The World Wide Web Consortium (W3C) has defined two standard languages for querying XML data: XPath and XQuery. XPath allows selecting XML node sets via tree traversal expressions. XQuery is an extension of XPath conceived to integrate multiple XML sources. We have an idea to improve the XQuery to be able to include the imprecise or incomplete values in terms of user's criteria.

As a consequence, this project proposes a way to extend XQuery language as providing a more flexible XQuery language with fuzzy set and implement an interpreter to execute the extended XQuery queries by using the GPFCSP concept.

2. Related Works

After we reviewed the literatures, we can summarize that there are three directions which research on fuzzy logic into XML technology at this time. The first is to use fuzzy logic in XML databases that uses of uncertain and imprecise values to define in database structures. The researchers are trying to define fuzzy XML model that will rely on existing database architecture. Definition of uncertain values in databases is achieved by reserving a part of a database for a fuzzy meta model that models fuzzy membership functions. Using an unconstrained set of membership functions would be difficult to implement. That is why authors define finite set of fuzzy membership function types that is enough for most uses. Besides, this type of papers will add the elements of fuzzy logic into the SQL query languages [1][2][3], and XQuery [4][5][6][7][8]. Authors introduce elements of syntax that allow the use of fuzzy elements with XML stored in a database. Additionally, some papers define extensions related to store procedures and triggers [9]. However, we rarely found the papers that implement the system in this direction.

The second is the use of fuzzy logic in XML documents. This approach divides into two parts – defining uncertainty of document structure and defining uncertainty values in XML elements and attributes. If an application needs to use XML documents, it is necessary that the structure of these documents is revealed. Since XML documents are developed together with other parts of a software system, introducing a new predefined structure of documents would cause problems with applications that read them. This is only an example of a problem that can be solved by introducing fuzzy logic to document structure. Some research defined query languages, typically by extending query languages like XPath [10][11], and XQuery [12].

The XML documents can have the uncertainty values. Sometimes a user is not able to precisely map real world values to XML document values. Classic logic demands precise definition of its elements. It is not able to work with values not defined in this way. However, values used in the real world are usually not precisely defined. While defining fuzzy XML document model, most often XML Schema and DTD Schema are used. Extensions of XML Schema language represent dominant way of defining fuzzy XML elements, because of its advantages over DTD. The example of papers that used the XML Schema are [13], [14] and [15] whereas [16] used DTD Schema. Sometimes even graph schema is used. Authors in this type of papers define syntax and implement fuzzy query language interpreters.

The third is use of fuzzy logic with XML for creating the modeling. The authors create the tools that extend the expressive power of UML with fuzzy logic. These tools give a foundation to create a

fuzzy XML based information system. Using these tools, users can develop XML models with fuzzy logic elements. The tools can contain possibilities to map fuzzy XML model to objects or database tables [17].

This project focuses on the first direction that applies fuzzy logic in XML databases. Our approach was inspired by the research work of Škrbić et al. [3] that proposed an extension of SQL with fuzzy capabilities called PFSQL (Prioritized Fuzzy Structured Query Language). In contrast, our research has been shifted to usage of XML technology. Another work that presents the similar approach as this project is Panić et al. [8]. They proposed the fuzzy XML and fuzzy XQuery extension which used GPFCSP concept to calculate the membership degree like in our work. However, the main difference between Panić's work and our work is that Panić's implementation used .NET framework, MATLAB and Microsoft SQL Server database, whereas our approach used Java programming language to implement the new interpreter independent of MATLAB with eXist-db - native XML database.

3. GPFCSP (Generalized Prioritized Fuzzy Constraint Satisfaction Problem)

The concept of Constraint Satisfaction Problem (CSP) is the problem defined as a set of objects which state must satisfy a number of constraints or limitations. The CSP can be extended to the FCSP (Fuzzy Constraint Satisfaction Problem) by modeling constraints as fuzzy sets over a particular domain. The PFCSP (Prioritized Fuzzy Constraint Satisfaction Problem) [18] is a type of FCSP that introduces the notion of priority. In this way, the value of constraint with the highest priority has the largest impact on the result. However, PFCSP only describes the use of the conjunction of the constraints. Takači et al. [19] generalized the PFCSP to the GPFCSP by adding the possibility to use disjunction and negation whereas the definition of the GPFCSP is much the same as the definition of the PFCSP.

Formal definition of GPFCSP may be found in [20]. Here we present a theorem that describes one practically usable GPFCSP system.

Theorem the following system (X, D, C^f, ρ, g, $\wedge$, $\vee$, $\neg$, $\Diamond$) where

1. $X = \{x_i \mid i = 1, 2, \ldots, n\}$ is a set of variables,

2. $D = \{d_i \mid i=1, 2, \ldots, n\}$ is a set of domains. Every domain d_i is a set that contains possible values of variable $x_i \in X$,

3. C^f is a set of fuzzy c, that is,

$$C^f = \{ \quad R_i^f \mid \mu_{R_i^f} : d_{i1} \times \ldots \times d_{ik_i} \longrightarrow [0, 1], i = 1, \ldots, m, 1 \leq k_i \leq n\}$$

where R_i^f denotes the set of constraint variables

4. $\rho : C^f \longrightarrow [0,\infty)$ is the priority of each constraint,

5. g: $[0,\infty) \times [0,1] \longrightarrow [0,1]$ is the global satisfaction degree,

6. $\wedge = T_L$,

7. $\vee = S_L$,

8. $\neg = 1-x$,

9. $\Diamond(x_i, c_i) = S_p(x_i, 1-p_i)$, $p_i = \rho(C_i)$ represents its priority.

10. v_X is a simultaneous valuation $v_X(x_1,\ldots,x_n)$, $x_i \in d_i$ of all variables in X.

is a GPFCSP. The global satisfaction degree of a valuation v_X for a formula F is obtained in the following way:

$$\alpha_F(v_X) = F\{\Diamond(v_{x_i}, \frac{\rho(R^f)}{\rho_{max}}) \mid R^f \in C^f\},$$

where C^f is the set of constraints of formula F, $\rho_{max} = \max\{\rho(R^f), R^f \in C^f\}$.

In a way, GPFCSP systems can be made similar to XQuery FLWOR (For-Let-Where-Order by-Return) clause. Basic structure of the FLWOR clause consists of *for*/*let* and *where* constructs. Variables that follow after the *for* and *let* keywords can be viewed as GPFCSP variables with associated domains. The *where* clause contains sequence of constraints connected with logical operators in much the same way as in GPFCSP.

4. System Implementation

This section contains details about the system implementation that is divided into seven subsections: System Architecture, Fuzzy XQuery Interpreter, Fuzzy XQuery EBNF grammar, Fuzzy values, Fuzzy compatibility, Fuzzy ordering and Graphic User Interface.

4.1 System Architecture

The aim of this research is to implement an interpreter that allows executing of fuzzy XQuery. We used Java programming language and defined the grammar of the fuzzy XQuery language with ANTLR (ANother Tool for Language Recognition) [21]. ANTLR is the tool for automatic generation of a lexical analyzer and a parser for the given EBNF Grammar. We chose eXist-db [22] as a Native XML database because it provides a pluggable module interface that allows extension modules to be easily developed in Java. These extension modules have full access to the eXist database for XQuery execution.

Our system architecture consists of three main components: *Fuzzy XQuery Interpreter*, *Compatibility Operation* and *Fuzzy ordering* as in figure 1.

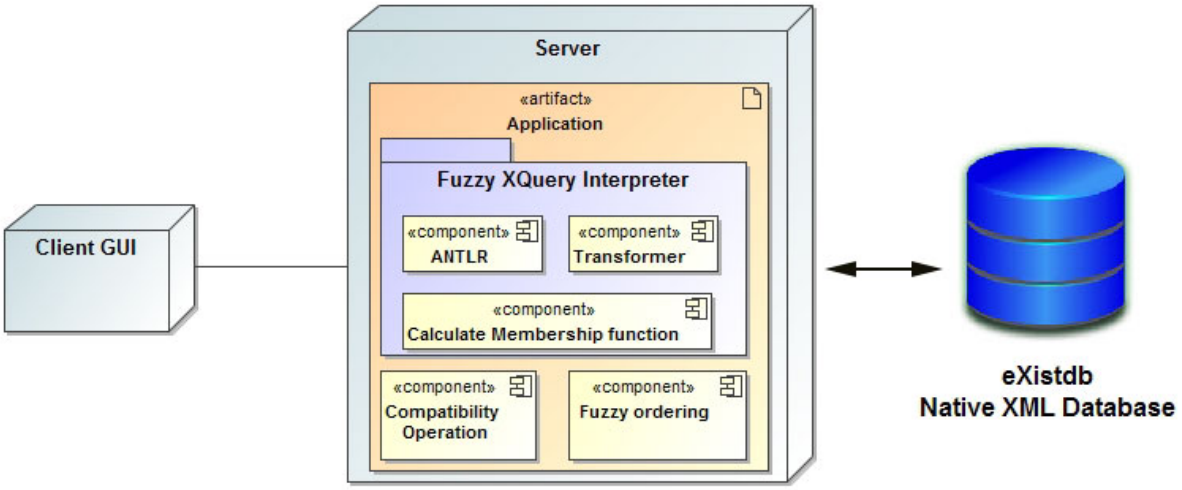


Figure 1: The system architecture of our approach

In the *Fuzzy XQuery Interpreter*, we defined the fuzzy XQuery grammar (see section 4.2: Fuzzy XQuery EBNF grammar) and calculated the global constraint satisfaction degrees of the result set (see section 4.4: Fuzzy XQuery Interpreter). The *Compatibility Operation* and *Fuzzy ordering* are the component for comparing two fuzzy sets. The *Compatibility Operation* (see section 4.5: Compatibility Operation) uses to compare two fuzzy sets with the equality (=) and inequality operator (!=). In the other hand, the *Fuzzy ordering* (see section 4.6: Fuzzy ordering) uses when the comparison operator is relational operators (<, <=, >, >=). Additionally, we defined the fuzzy values, which can be used in the fuzzy XQuery query, in XML document within our database (see section 4.3 Fuzzy values). The list of software used is shown in table 1.

Table 1: List of softwares

	Software	**Version**
Operating System	Windows 7	64 bit
Database	eXist-db	2.1
Web Server	Apache Tomcat	8.0
Tools	Java	Standard Edition Development Kit (Linux x64) 7u7
	Eclipse	IDE for Java EE Developers (Luna Packages)
	ANTLR	3.4 (Complete ANTLR 3.4 Java binaries jar)

4.2 Fuzzy XQuery EBNF grammar

We extend some of the standard XQuery EBNF 1.0 notation in the *where* clause of the FLWOR statement with the keywords *priority* and *threshold*. The extended fuzzy XQuery syntax is described by using the EBNF notation as in figure 2. The PriorityExpr is an expression with the keyword *priority* which has the effect to define the level of influence criteria on the result. The ThresholdExpr, like PriorityExpr, is an expression with threshold that applies the concept of α-cut to reject those results that are under the specified threshold value. Both values of *priority* and *threshold* are in the unit interval [0, 1]. If there is no priority, it will assume that the value is 1. On the other hand, if the query does not specify the threshold, the value is 0.

```
WhereClause      ::=  "where" ExprSingle (ThresholdExpr)?
ExprSingle       ::=  OrExpr
ThresholdExpr    ::=  "threshold" DecimalLiteral
OrExpr           ::=  AndExpr ("or" AndExpr)*
AndExpr          ::=  ComparisonExpr ("and" ComparisonExpr)*
ComparisonExpr   ::=ValueExpr((GeneralComp)ValueExpr)
ValueExpr        ::=FuzzyExpr (PriorityExpr)?
GeneralComp      ::= '='|'!= '|'<'|'<='|'>'|'>='
FuzzyExpr        ::'#' 'ling' '('Qname')' '#'
                 | '#' 'tri' '('leftoffset','max','rightoffset')' '#'
                 | '#' 'trap' '('leftoffset','leftmax','rightmax','rightoffset')' '#'
                 | '#' 'interval' '('leftoffset','rightoffset')' '#'
                 | '#' 'fs' '('type','leftoffset','rightoffset',)' '#'
PriorityExpr     ::="priority" DegreeLiteral
```

Figure 2: The EBNF notation of extended XQuery

Moreover, we defined the fuzzy expression in grammar as in FuzzyExpr. There are 5 types of fuzzy value as follows: Linguistic label, Triangle, Trapezoidal, Interval and Fuzzy shoulder.

1) Linguistic label-ling (Qname)

Fuzzy value is a linguistic label with name given by *Qname.*

2) Triangle-triangle (leftoffset, max, rightoffset)

Fuzzy value is a triangular fuzzy number with maximum in *max* and left and right offsets denoted by *leftoffset* and *rightoffset*.

3) Trapezoidal-trapezoid (leftoffset, leftmax, rightmax, rightoffset)

Fuzzy value is a trapezoidal fuzzy number with maximum on the [*leftmax, rightmax*] interval and left and right offsets denoted by *leftoffset* and *rightoffset*.

4) Interval-interval (left, right)

Fuzzy value is an interval with left and right boundaries given by *left* and *right*.

5) Fuzzy shoulder-fs (type, leftoffset, rightoffset)

Fuzzy value is a fuzzy shoulder which has two types (left shoulder and right shoulder) with left and right boundaries given by leftoffset and rightoffset.

Suppose that user wants to retrieve information about students that have height more than 170 cm. and *young*. But the property height is more important than *young* which has the priority with 0.5. For this purpose, the user can define the fuzzy XQuery with *priority* and *threshold* as in figure 3. The query also contains the threshold clause that limits the results and removes the results with the fuzzy satisfaction degree smaller than 0.6.

```
for $x in document("students.xml")//student
where $x/height > 170 AND $x/age = #ling(young) priority 0.5
threshold 0.6
return  $x/name
```

Figure 3: An example of fuzzy XQuery

4.3 Fuzzy values

We store fuzzy data (the linguistic labels, the linguistic variables, and the possibility distributions) in an XML document and use them to describe membership functions of fuzzy values. A linguistic variable (such as age) has a range of values and at least one linguistic label. The linguistic label (such as young) describes the standard type of fuzzy values (triangle, trapezoidal, interval and fuzzy shoulder) and the values of distribution in a numeric data type with tag <offset>. The DTD (Document Type Definition) in figure 4 shows the structure of XML elements used for definition of fuzzy values.

```
<!DOCTYPE linguistics [
        <!ELEMENT linguistics (linguistic+ )>
        <!ELEMENT linguistic (name, (triangular|trapezoidal|interval|leftshoulder|rightshoulder))>
        <!ATTLIST linguistic var NMTOKEN #REQUIRED >
        <!ELEMENT name (#PCDATA ) >
        <!ELEMENT triangular (leftOffset, maxOffset, rightOffset)>
        <!ELEMENT trapezoidal (leftLowerOffset, leftUpperOffset, rightUpperOffset,
        rightLowerOffset)>
        <!ELEMENT interval (leftOffset, rightOffset) >
        <!ELEMENT leftshoulder (leftOffset, rightOffset)>
        <!ELEMENT rightshoulder (leftOffset, rightOffset)>
        <!ELEMENT leftOffset (#PCDATA)>
        <!ELEMENT rightOffset (#PCDATA)>
        <!ELEMENT maxOffset (#PCDATA)>
        <!ELEMENT leftLowerOffset (#PCDATA)>
        <!ELEMENT leftUpperOffset (#PCDATA)>
        <!ELEMENT rightUpperOffset (#PCDATA)>
        <!ELEMENT rightLowerOffset (#PCDATA)>
        <!ELEMENT UpperOffset (#PCDATA)>
        <!ELEMENT LowerOffset (#PCDATA)>
]>
```

Figure 4: DTD for defining linguistic variables

For example (as in figure 5), the linguistic variable “age” has a value of linguistic label “young” and it is corresponding to the fuzzy shoulder distribution (left shoulder) with the offset of 20 and 25, respectively.

```
<?xml version=“1.0”?>
<!--define linquistic variables: age-->
<linguistics>
   <linguistic var="age">
      <name>young</name>
      <leftshoulder>
            <leftOffset>20</leftOffset>
            <rightOffset>25</rightOffset>
      </leftshoulder>
   </linguistic>
</linguistics>
```

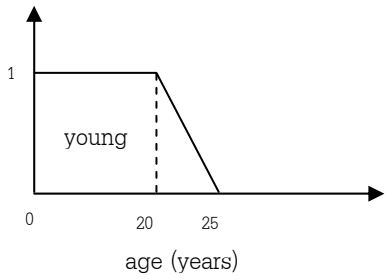


Figure 5 The definition of linguistic variable “age”

4.4 Fuzzy XQuery interpreter

Now let us turn our attention to how fuzzy XQuery interpreter’s work. Figure 6 shows three main steps for executing the fuzzy XQuery: 1) check syntax; 2) transform and 3) calculate the membership degree as follows:

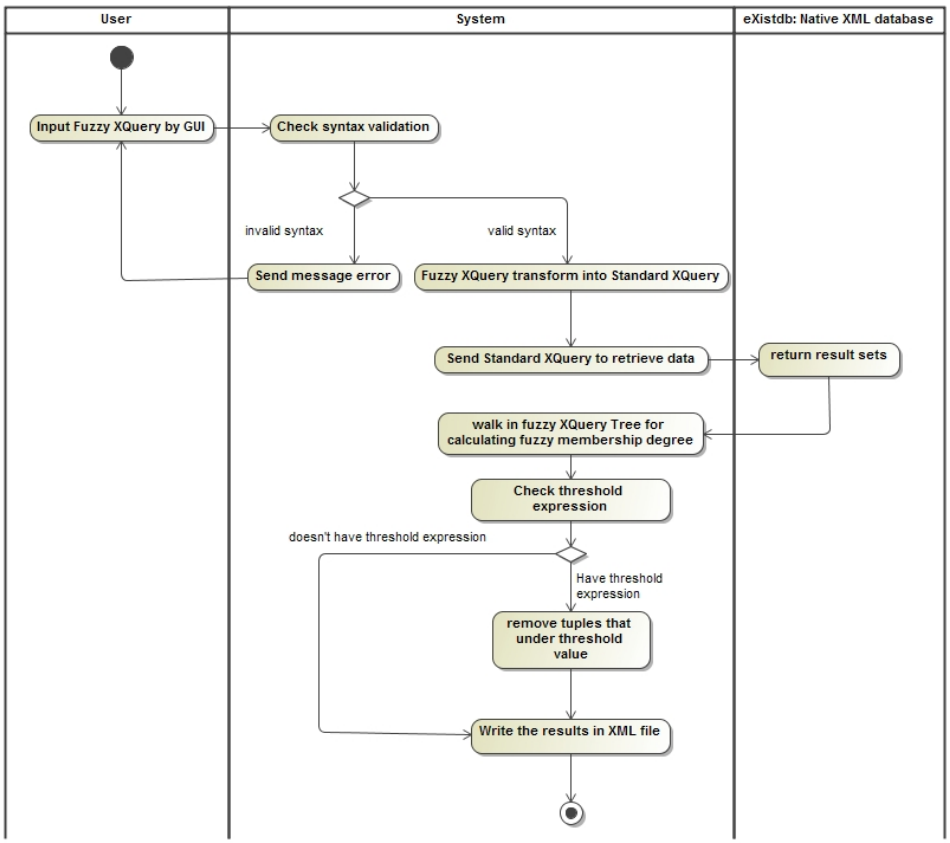


Figure 6 Activity diagram for executing fuzzy XQuery

Step 1: Check syntax

When a user inputs a fuzzy XQuery query, the system first checks and validates its syntax by following the EBNF (Extended Backus Normal Form) grammar (as can be seen in figure 2 and appendix A). In this step, we use the ANTLR to generate the parser and create an AST (Abstract Syntax Tree).

Step 2: Transform

The fuzzy XQuery is transformed to a classical XQuery query by extracting the fuzzy parts from it. This step uses the *Transformer* component which the system will check for fuzzy element in the query, eliminate the fuzzy constraint and move the fuzzy expression from the fuzzy XQuery.

At this point, let us consider how to transform the fuzzy XQuery to standard XQuery. Firstly, the system checks and validates the fuzzy XQuery's syntax by following the EBNF grammar. The ANTLR will generate the AST. For example, if we have the fuzzy XQuery as follows:

"for $x in doc("db/data/student.xml")/students/student
where $x/tall > 170 and $x/age = #ling(young)# priority 0.5 threshold 0.6
return $x/name"

We will have the AST as in figure 7. It is noticeable that the *priority* and *threshold* will not be in the tree and the *fuzzyexpr* will be replace by the FUZZY token

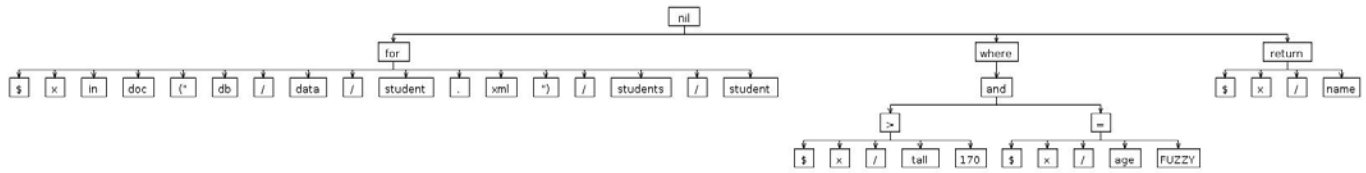


Figure 7 The created AST by ANTLR

Secondly, we traverse the AST and extract only the *whereclause* subtree (as in figure 8) and delete the FUZZY node from the query.

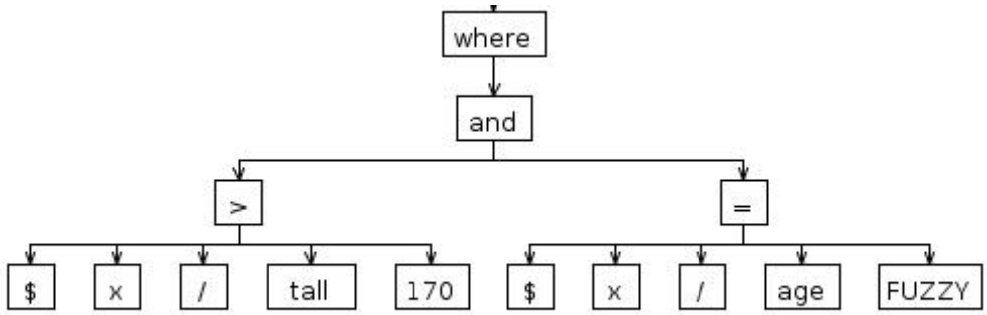


Figure 8 The *whereclause* subtree

There are 4 steps for deleting the FUZZY node from the *whereclause* subtree (as shown in figure 9):

❶Search the FUZZY tokens from the child node. If found, delete the branch of the tree that has the FUZZY token.

❷If the parent of the branch (which was deleted in step 1) is the conjunction token (AND or OR), delete the conjunction token and put the sibling branch to replace the conjunction token.

❸ Traverse the tree and check for every FUZZY tokens in the tree.

❹we will have the new *whereclause* tree as shown in figure 10.

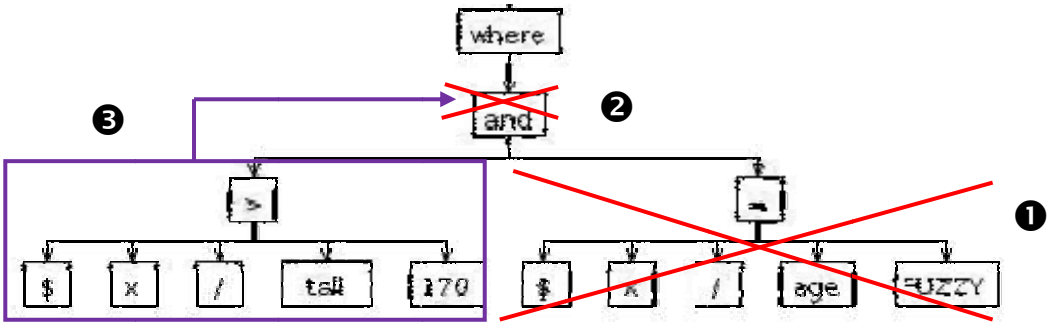


Figure 9 The steps used to delete the FUZZY node from *whereclause* subtree

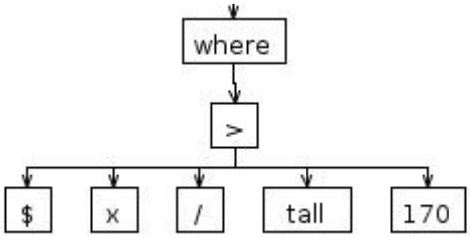


Figure 10 The *whereclause* subtree after the FUZZY token was deleted

Thirdly, after we have the new *whereclause* subtree, we walk through the *whereclause* subtree again and write it to standard XQuery with inorder traversal. Inorder walk traverses the left subtree, visits the root and finally traverses the right subtree. In figure 11, we show how to traverse the tree as inorder walk: I, traverse the left subtree ($, x, /, tall). II, Visit the root node (>). III, traverse the right subtree (170).

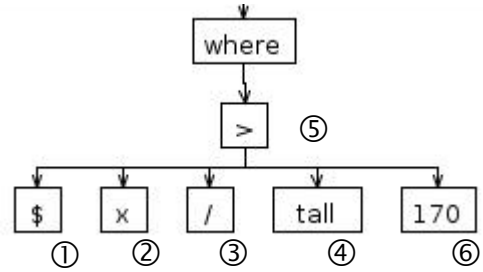


Figure 11 The inorder walk in *whereclause* subtree

Finally, from this example, we have the tree as in figure 12 and standard XQuery as follows:

```
for $x in doc("db/data/student.xml")/students/student
where $x/tall > 170
return $x/name
```

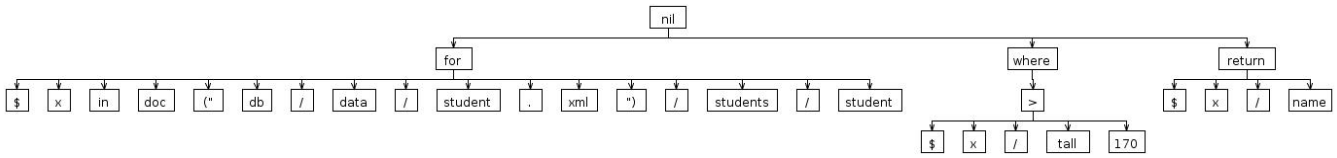


Figure 12 The tree after the fuzzy node was deleted

Step 3: Calculate the membership degree

The system will retrieve the data from the standard XQuery in step 2. When the database returns the results, they are again interpreted having original query in mind. Fuzzy elements are to be interpreted using known GPFCSP concept. This step uses the *CalculateMembershipFn*, *Compatibility Operation* [23] and *Fuzzy ordering* component [24].

We now explain in detail how to use the GPFCSP concept for calculating the global constraint satisfaction degree (α) for every tuples in a result set by using algorithm as can be seen below.

Algorithm used for calculating the global constraint satisfaction degree (α)

1. Walk the fuzzy XQuery tree.
2. Find the *whereclause* subtree.
3. Walk the *whereclause* subtree until the end of subtree,
   - 3.1 check the type of current node
     - 3.3.1 If the current node = 'AND', then $\alpha = T_L$(LeftBranch,RightBranch)
     - 3.3.2 If the current node = 'OR', then $\alpha = S_L$(LeftBranch,RightBranch)
     - 3.3.3 If the current node is '=', '!=', '>', '>=', '<', '<=', then walk

```
the child node
A. If the child node = '/', then get the variable of condition
   after the '/' node
B. If the child node ='ling'
   a) Read the linguistic variable after 'ling' node
   b) Take the distribution of the linguistic variable in step a)
   from DB
   c) Get the value of variable in step A from DB
      i) If the value is numeric data, then α is the value of
       corresponding membership function in the given point
      ii) Else if the value is fuzzy value
          • Check type of the operator
            o If the operator is  = or != then call Compatibility
              Function
                ▪ If the operator is =, then α = Compatibility value
                ▪ If the operator is !=, then α = 1-Compatibility
                  value
            o Else if the operator is >,>=,<,<= then call Fuzzy
              Ordering Function
                ▪ If the operator is < or <=, then α = Fuzzy
                  ordering value
                ▪ If the operator is > or >=, then α = 1- Fuzzy
                  ordering value
   d) If it has priority expression, then α = SP(α, 1-priority
      value)
C. If the child node is 'tri', 'trap', 'interval' or 'fs'
   a) Read the offsets of the 'tri', 'trap', 'interval' or 'fs'
      node
   b) Get the value of variable in step A from DB
   c) Check the type of the operator
     i) If the operator is  = or != then call Compatibility
        Function
          • If the operator is =, then α = Compatibility value
          • If the operator is !=, then α = 1-Compatibility value
     ii) Else if the operator is >,>=,<,<= then call Fuzzy Ordering
         Function
          • If the operator is < or <=, then α = Fuzzy ordering
            value
          • If the operator is > or >=, then α = 1- Fuzzy ordering
            value
   d) If it has priority expression, then α = SP(α, 1-priority
      value)
```

From the algorithm, when node is conjunction 'AND', we calculate the α by calling the Łukasiewicz triangular norm (T_L) function. Similar as the disjunction 'OR' node, we use Łukasiewicz triangular conorm (S_L). If the fuzzy XQuery has priority expression, we use the triangular product conorm (S_P) to aggregate with priority value. When node is the operator = or != , a *Fuzzy Compatibility* function is called. The *Fuzzy Compatibility* function is explained in section 4.5. When node is the operator >, >=, < or <= , a *Fuzzy Ordering* function is called. The *Fuzzy Ordering* function is explained in section 4.6

After calculating the α for every tuples, if the fuzzy XQuery has threshold expression, the system will remove the tuples which have the α under the threshold value.

Additionally, we illustrate how to calculate the gobal constraint with an example in our proposed

papare that will be published in the Special Issues of Journal Teknologi (see appendix B: ADVCIT Paper).

4.5 Fuzzy Compatibility

In fuzzy XQuery statements, variables can be fuzzy or non-fuzzy value. Normally, different types of values cannot be compared directly. Therefore, it is necessary to implement fuzzy compatibility calculation to solve this problem. In 2013, Škrbić [25] proposed the equation of compatibility of fuzzy set A to the fuzzy set B is given below.

$$C_{A,B} = \frac{P(A \cap B)}{P(A)} \quad \text{(E1)}$$

$P(A \cap B)$ is the area of intersection between the two fuzzy sets and P(A) is the area of the source fuzzy set A. Compatibility value is a number that varies from 0 to 1. Zero means incompatible, and one means fully compatible. According to equation (E1), the fuzzy compatibility calculation contains 3 steps:

1) Calculate intersection area
2) Calculate size of obtained intersection area
3) Calculate compatibility value

To obtain intersection area of two fuzzy sets, the edge equations of each fuzzy sets will be compared as well as their boundaries. The output of the step 1 is a coordinates of the intersection area that will be used to calculate size of the intersection area in step 2.

In step 2, the cyclic polygon calculation proposed by Pak [26] will be used to calculate size of the intersection area. This method uses coordinates of a polygon for the area calculations. The area is calculated by the following equation:

$$Area = \left| \frac{(x_1 y_2 - x_2 y_1) + (x_2 y_3 - x_3 y_2) + \cdots + (x_n y_1 - x_1 y_n)}{2} \right| \quad \text{(E2)}$$

In step 3, a compatibility value can be obtained by equation (E1) using size of the intersection area provided by step 2 and size of area of the source fuzzy set that can be calculated by the equation (E2).

The idea of fuzzy compatibility has been proposed in International Journal of Machine Learning and Computing (IJMLC) (see appendix C). To prove the correctness of the compatability calculation

process, 360 compatibility cases are generated for experimental in this paper. The experimental results show that the proposed algorithms for compatibility calculations can handle various types of intersections between any two fuzzy sets and can be used for the fuzzy logic enriched XQuery lanaguge.

4.6 Fuzzy Ordering

When the relational operators are included in the query, it is necessary to provide means for comparison between fuzzy sets. These fuzzy relational operators are typically used in two fuzzy sets comparison case, but can also be used with some aggregate functions like MIN, MAX, and SUM. In 2008, fuzzy ordering is proposed by Bodenhofer [27]. The proposed method is given below:

$$A \preccurlyeq_I B \Leftrightarrow (LTR(A) \supseteq LTR(B) \bigwedge RTL(A) \subseteq RTL(B)) \quad \text{(E3)}$$

The inclusion $LTR(A) \supseteq LTR(B)$means that the left flank of A is to the left of the left flank of B while $RTL(A) \subseteq RTL(B)$ means that the right flank of A is to the left of the right flank of B.

Considering fuzzy orderings above, the fuzzy ordering calculation can be determined by considering horizontal positions of comparing fuzzy sets. If the assertion (E3) is fulfilled in both conditions, the fuzzy ordering value is true or 1. Otherwise, the operation returns false or 0. From assertion (E3) can be concluded that if only one condition is satisfied, it means that fuzzy sets cannot be compared - incomparable case. In this case, the fuzzy ordering operation will return incomparable or 0.5.

Another incomparable case is the comparison of fuzzy sets having different heights. However, Skrbic and Rackovic proposed an idea to eliminate this problem in [26]. Fuzzy set A′ is introduced as:

$$\mu_{A'} = \begin{cases} 1, \mu_A(x) = h(A) \\ \mu_A(x), otherwise \end{cases} \quad \text{(E4)}$$

In this way, fuzzy relational operator $\leq_F$ is introduced by:

$$A \leq_F B \Leftrightarrow A' \leq'_F B' \quad \text{(E5)}$$

From equation (E4) and assertion (E5), the fuzzy ordering that can compare two fuzzy sets with different heights defined by:

$$A \leq'_F B \Leftrightarrow (LTR(B) \subseteq LTR(A) \wedge RTL(A) \subseteq RTL(B)) \quad \text{(E6)}$$

In the same way as with operators $<$ and $>$ on crisp domain, other relational operators, like $<_F$ and $>_F$ can be derived using the $\leq_F$ order.

The concept of fuzzy ordering has been proposed in International Conference on Information Society and Technology (ICIST 2015). (see appendix D). The proposed fuzzy ordering process is evaluated with 360 fuzzy ordering cases. The experimental results show that the proposed algorithms are capable of calculating fuzzy ordering values with various types of fuzzy values correctly.

5. Graphical User Interface (GUI)

We developed a GUI that allows the fuzzy XQuery query to execute on our application. The GUI has two parts: ①*Input* and ②*Output* as shown in Figure 13 and 14. The *Input* is on the top part of the page which a user can add the fuzzy XQuery query in the text box and click on the *CALCULATE* button to submit the query into our system. After that the result will be shown in the *Output* part. There are two views of the output: table view and XML view. In the table view, the results are shown in the Results tab (see Figure 13). The table of results has six columns: ID, name, GPA, age, height and alpha. The first five columns are the data from the database and the last column is the global constraint satisfaction degree (alpha) of that record. If the user clicks on the *XML Results* tab, the same results as in the *Results* tab are shown in the XML view (see Figure 14).

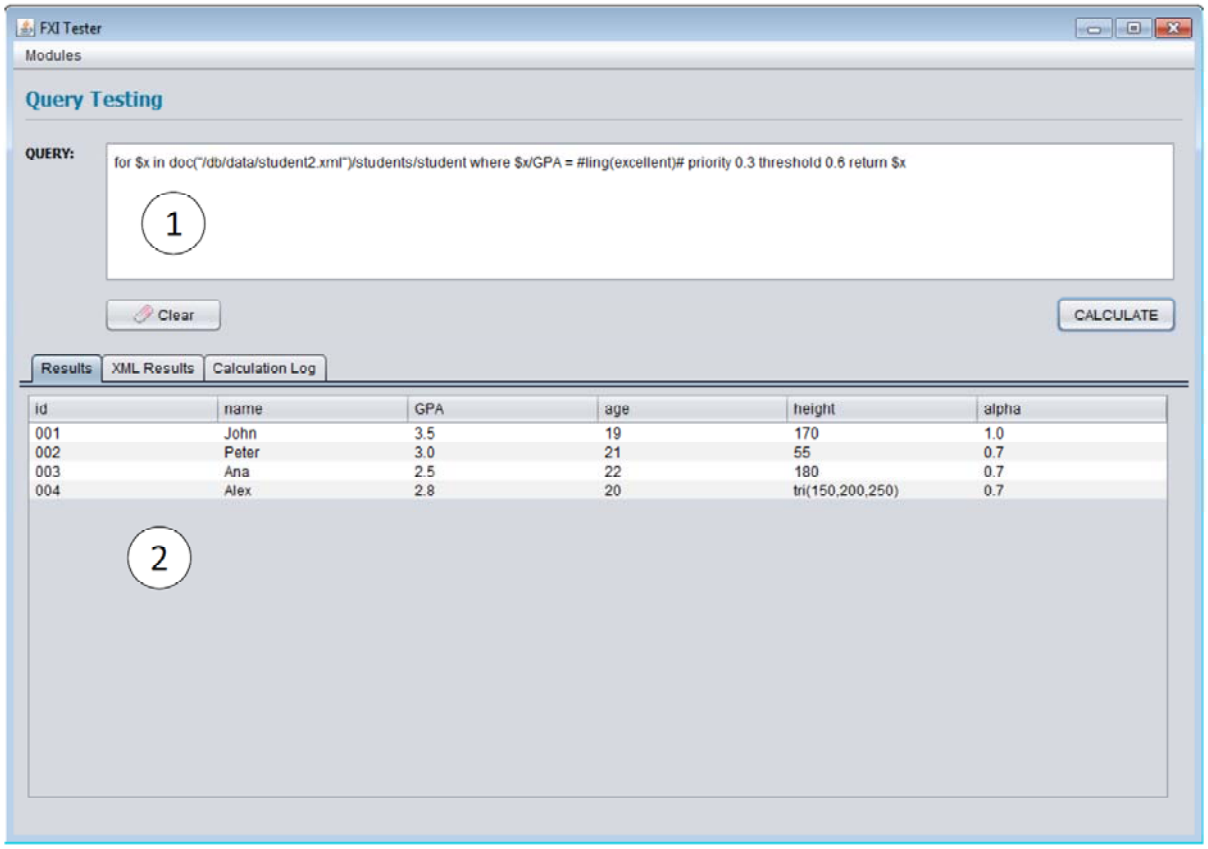


Figure 13 Interface with output in table view

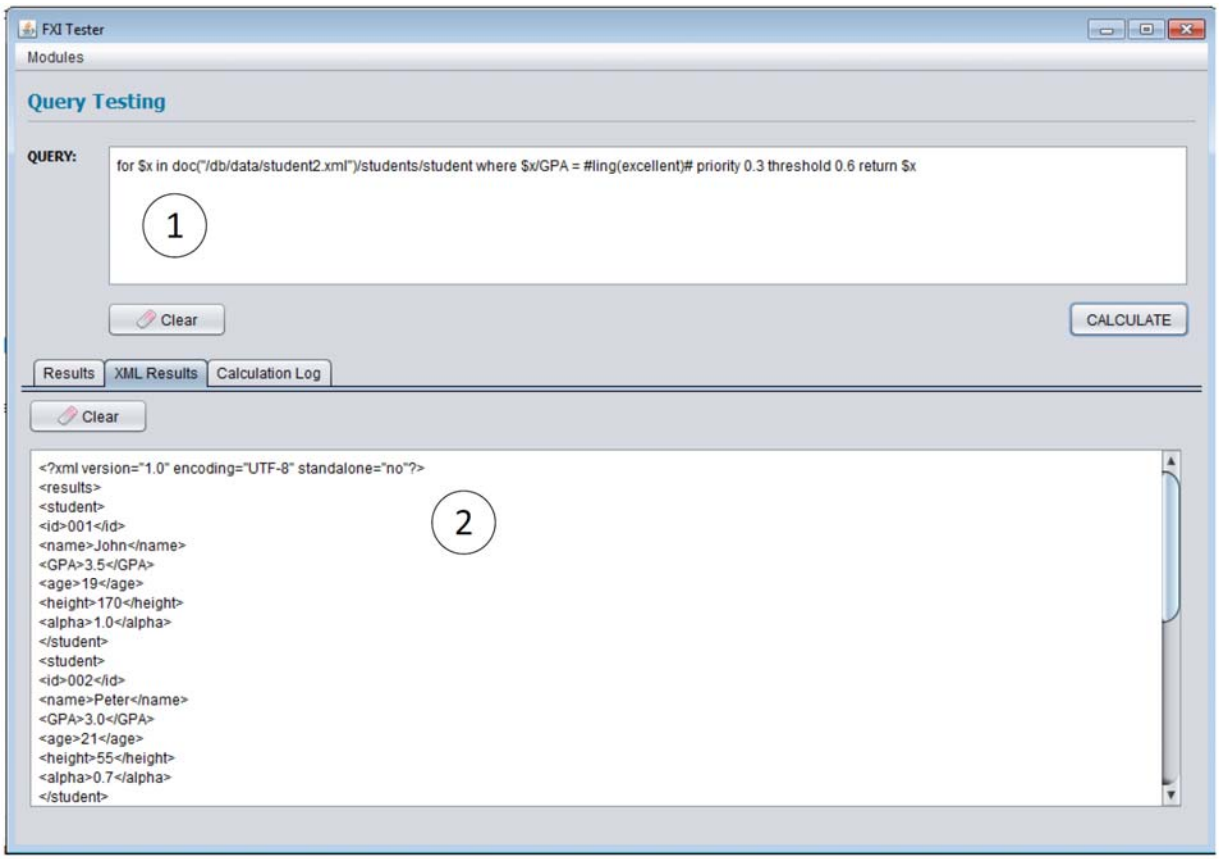


Figure 14 Interface with output in XML views

6. Research Publications

This project has accepted for publications as follows:

6.1 The paper entitled "A GPFCSP Based Fuzzy XQuery Interpreter" has been accepted by 2nd Advancement on Information Technology International Conference (ADVCIT 2015), Thailand and will be published in the Special Issues of Journal Teknologi (Indexed by Elsevier: SCOPUS) (see appendix B: ADVCIT Paper).

6.2 The paper entitled "Polygon Intersection Based Algorithm for Fuzzy Set Compatibility Calculations" has been published by International Journal of Machine Learning and Computing (IJMLC), Vol. 6, No. 1, pp. 32 – 35, February 2016. (see appendix C: ICIT Paper).

6.3 The paper entitled "Fuzzy ordering implementation applied in fuzzy XQuery" accepted by 2015 International Conference on Information Society and Technology (ICIST 2015), Kopaonik, Serbia (see appendix D: ICIST Paper).

7. Conclusion

We present the extension of the XQuery language with fuzzy set that would allow to adding the priority and threshold in the query, incorporate fuzzy information into XML documents and define the structure of fuzzy XML documents by using DTD schema. The interpreter is implemented by using Java language and the GPFCSP concept for execution of the fuzzy XQuery queries. Moreover, the system is based on a native XML database: eXist-db.

References

[1] C. D. Barranco, J. R. Campana, and J. M. Medina, “Towards a XML Fuzzy Structured Query Language,” in *Proceedings of the Joint 4th Conference of the European Society for Fuzzy Logic and Technology and the 11th Rencontres Francophones sur la Logique Floue et ses Applications*, Barcelona, Spain, 2005, pp. 1188–1193.

[2] R. D. Rodrigues, A. J. O. Cruz, and R. T. Cavalcante, “Aliança: A proposal for a fuzzy database architecture incorporating XML,” Fuzzy Sets Syst., vol. 160, no. 2, pp. 269 – 279, 2009.

[3] S. Škrbić and M. Racković, “PFSQL: a fuzzy sql language with priorities,” in *Proceedings of the 4th International Conference on Engineering Technologies (ICET 2009)*, 2009, pp. 119–125.

[4] E. J. Thomson Fredrick and G. Radhamani, “Fuzzy logic based XQuery operations for native XML database systems,” *International Journal Database Theory and Application*, vol. 2, no. 3, pp. 13–20, September 2009.

[5] E. J. Thomson Fredrick and G. Radhamani, “A GUI based tool for generating XQuery and fuzzy XQuery,” *International Journal of Computer Applications Database Theory and Application*, vol. 1, no. 17, pp. 54–58, 2010.

[6] E. J. Thomson Fredrick and G. Radhamani, “Information retrieval using XQuery processing techniques,” *International Journal of Database Management Systems (IJDMS)*, vol. 3, no. 1, pp. 50–58, February. 2011.

[7] E. J. Thomson Fredrick and G. Radhamani, “Fuzzy integrity constraints for native xml database,” vol. 9, issue 2, no. 3, p. 466–471, March 2012.

[8] G. Panić, S. Škrbić, and M. Racković, “Fuzzy xml and prioritized fuzzy xquery with implementation,” Journal of Intelligent and Fuzzy Systems, vol. 26, no. 1, pp. 303–316, 2014.

[9] Ying Jin and S. Shidlagatta, “A framework of fuzzy triggers for XML database systems,” in *Proceedings of the 10th IEEE international conference on Information Reuse & Integration*, Las Vegas, USA, 2009, pp. 296–299.

[10] E. Damiani, S. Marrara, and G. Pasi, “FuzzyXPath: Using Fuzzy Logic an IR Features to Approximately Query XML Documents,” in *Foundations of Fuzzy Logic and Soft Computing*, vol. 4529, P. Melin, O. Castillo, L. Aguilar, J. Kacprzyk, and W. Pedrycz, Eds. Springer Berlin Heidelberg, 2007, pp. 199–208.

[11] A. Campi, E. Damiani, S. Guinea, S. Marrara, G. Pasi, and P. Spoletini, “A Fuzzy Extension of the XPath Query Language,” *Journal of Intelligent Information System*, vol. 33, no. 3, pp. 285–305, Dec. 2009.

[12] M. Goncalves and L. Tineo, “Fuzzy XQuery,” in *Soft Computing in XML Data Management*, vol. 255, Z. Ma and L. Yan, Eds. Springer Berlin Heidelberg, 2010, pp. 133–163.

[13] B. Oliboni and G. Pozzani, “Representing Fuzzy Information by Using XML Schema,” in *Proceedings of the 19th International Conference on Database and Expert Systems Application*, 2008, pp. 683–687.

[14] H. Wang, Z. M. Ma, L. Yan, and J. Cheng, “A Unified Formalism for Fuzzy Data Types Representation,” in *Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 2008, pp. 167 – 171.

[15] L. Yan, Z. M. Ma, and J. Liu, “Fuzzy Data Modeling Based on XML Schema,” in *Proceedings of the 2009 ACM Symposium on Applied Computing*, 2009, pp. 1563–1567.

[16] C. Tseng, W. Khamisy, and T. Vu, “Universal fuzzy system representation with XML,” *Comput. Stand. Interfaces*, vol. 28, no. 2, pp. 218 – 230, 2005.

[17] Z. M. Ma and L. Yan, “Fuzzy XML data modeling with the UML and relational data models,” *Data Knowl. Eng.*, vol. 63, no. 3, pp. 972 – 996, 2007.

[18] D. Dubois, H. Fargier, and H. Prade, “Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty,” Applied Intelligence, vol. 6, 1996, pp. 287–309.

[19] A. Takači, S. Škrbić, and A. Perović, "Generalized Priority Constraint Satisfaction Problem," in the 7th Serbian-Hungarian Joint Symposium Intelligent Systems and Informatics, Subotica, Serbia, 2009, pp. 145–148.

[20] S. Škrbić, M. Racković, and A. Takači, "Prioritized fuzzy logic based information processing in relational databases," Knowledge-Based Systems, vol. 38, pp. 62-73, January 2013.

[21] T. Parr. (2012) ANTLR v3. [Online]. Available: http://www.antlr.org/.

[22] eXistdb project. (2014) eXistdb. [Online]. Available: http://existdb.org/exist/apps/homepage/index.html.

[23] Sukpisit, S., Kansomkeat, S., Sae Ueng P., Thadadech, A., and Skrbic S. “Polygon Intersection Based Algorithm for Fuzzy Set Compatibility Calculations,” International Journal of Machine Learning and Computing (IJMLC), Vol. 6, No. 1, pp. 32 – 35, February 2016.

[24] S. Kansomkeat, S. Sukpisit, A. Thadadech, P. S. Ueng, and S. Škrbic,´"Fuzzy ordering implementation applied in fuzzy XQuery," in Proc. Of International Conference on Information Society and Technology (ICIST 2015), Kopaonik, Serbia, 2015, pp. 443-448.

[25] S. Škrbić and M. Racković, *FUZZY DATABASES*. Faculty of Science, University of Novi Sad, 2013.

[26] I. Pak, "The area of cyclic polygons: Recent progress on Robbin's conjectures," Advances in Applied Mathematics 34, pp. 690-696, 2005.

[27] U. Bodenhofer, "Orderings of fuzzy sets based on fuzzy orderings part i: The basic approach," *Mathware & Soft Computing*, vol. 15, pp. 201–218, 2008.

Appendix A: Fuzzy XQuery EBNF grammar

```
grammar FuzzyXQueryFull;

options {
  language = Java;
  output=AST;
  ASTLabelType=CommonTree;
}

tokens{
	FUZZY;
	PRIORITY;
}

@lexer::header{
	package grammar;
}

@parser::header{
	package grammar;
}

querybody
	:	expr
	;
expr
	:	exprsingle (',' exprsingle)*
	;
exprsingle
	: flowrexpr
	|orexpr
	;
flowrexpr
	:(forclause|letclause)+  whereclause? orderbyclause? returnclause
	;
forclause
	:	'for'^ '$' varname typedeclaration? positionalvar? 'in' exprsingle
(',' '$' varname typedeclaration? positionalvar? 'in' exprsingle)*
	;
positionalvar
	:	'at' '$' varname
	;
letclause
	:	'let'^ '$' varname typedeclaration? ':=' exprsingle (',' '$' varname
typedeclaration? ':=' exprsingle)*
	;
whereclause
	:	'where' exprsingle (thresholdexpr)? -> ^('where' exprsingle)
thresholdexpr?
	;
orderbyclause
	:	('order' 'by' | 'stable' 'order' 'by') orderspeclist
	;
orderspeclist
	:	orderspec (',' orderspec)*
	;
```

```
orderspec
	:	exprsingle ordermodifier
	;
ordermodifier
	:	('ascending' | 'descending')? ('empty' 'greatest'| 'empty' 'least')?
('collation' uriliteral)?
	;
returnclause
	:	'return'^  '$' varname
	;
orexpr
	:	andexpr ( 'or'^ andexpr)*
	;
andexpr
	:	comparisonexpr ( 'and'^ comparisonexpr)*
	;
comparisonexpr
	:	rangeexpr ( generalcomp^ rangeexpr)?
	;
rangeexpr
	:valueexpr
	;
valueexpr
	: pathexpr
	| fuzzyexpr (priorityexpr)? ;

generalcomp
	:	'=' | '!=' | '<' | '<=' | '>' | '>='
	;
pathexpr
	:('/' relativepathexpr?)
	| ('//' relativepathexpr)
	| relativepathexpr
	;
relativepathexpr
	: stepexpr (('/' | '//')stepexpr)*
	;
stepexpr
	:filterexpr
	| axisstep
	;
axisstep
	:(reversestep | forwardstep) predicatelist
	;
forwardstep
	: (forwardaxis nodetest)
	| abbrevforwardstep
	;
forwardaxis
	:('child' '::')
	| ('descendant' '::')
	| ('attribute' '::')
	| ('self' '::')
	| ('descendant-or-self' '::')
	| ('following-sibling' '::')
	| ('following' '::')
	;
abbrevforwardstep
	:	'@'? nodetest
```

```
	;
reversestep
	:(reverseaxis nodetest) | abbrevreversestep
	;
reverseaxis
	: ('parent' '::')
	| ('ancestor' '::')
	| ('preceding-sibling' '::')
	| ('preceding' '::')
	| ('ancestor-or-self' '::')
	;
abbrevreversestep
	:'..'
	;
nodetest
	:nametest
	;
nametest
	:QNAME
	;
filterexpr
	:primaryexpr predicatelist
	;
predicatelist
	:predicate*
	;
predicate
	:'[' expr  ']'
	;
primaryexpr
	:literal
	|varref
	|parenthesizedexpr
	|contextitemexpr
	|functioncall
	|orderedexpr
	|unorderexpr
	;
literal
	:numericliteral
	|STRINGLITERAL
	;
varref
	: '$' varname
	;
varname
	:	QNAME
	;
parenthesizedexpr
	:'(' expr?')' -> expr?
	;
contextitemexpr
	:QNAME '.' QNAME
	;
orderedexpr
	:'ordered' '{' expr '}'
	;
unorderexpr
	:'unordered' '{' expr '}'
```

```
	;
functioncall
	:QNAME '("'(exprsingle (',' exprsingle)*)? '")'  //add " and "
	;
ncname
	:name
	;
singletype
	:	atomictype '?'?
	;
typedeclaration
	:	'as' sequencetype
	;
sequencetype
	:	('empty-sequence' '(' ')')
	;
occurrenceindicator
	:'?' |'*'|'+'
	;
atomictype
	:	QNAME
	;
uriliteral
	:	STRINGLITERAL
	;
fuzzyexpr
	: '#' 'ling' '('QNAME')' '#' -> FUZZY 'ling' QNAME
	| '#' 'tri' '(' leftoffset','max','rightoffset')' '#' -> FUZZY 'tri'
leftoffset max rightoffset
	| '#' 'trap' '('leftoffset',' leftmax ','rightmax ',' rightoffset ')' '#' -
> FUZZY 'trap' leftoffset leftmax rightmax rightoffset
	| '#' 'interval' '(' leftoffset ',' rightoffset ')' '#' -> FUZZY 'interval'
leftoffset rightoffset
	| '#' 'fs' '(' type ',' leftoffset ',' rightoffset')' '#' -> FUZZY 'fs'
type  leftoffset rightoffset
	;
max
	:numericliteral
	;
leftoffset
	:numericliteral
	;
rightoffset
	:numericliteral
	;
leftmax
	:numericliteral
	;
rightmax
	:numericliteral
	;
type
	: '1' | '0';

priorityexpr
	:  'priority' degreeliteral -> PRIORITY degreeliteral
	;
thresholdexpr
	: 'threshold' degreeliteral
```

```
	;
numericliteral
	:integerliteral
	|decimalliteral
	|degreeliteral
	;
integerliteral
	:DIGITS
	;
decimalliteral
	:(DIGITS '.' DIGITS)
	;
degreeliteral
	: '0.' DIGITS
	;
predefinedentityref	//145
	: '&' ('lt'|'gt'|'amp'|'quot'|'apos')';'
	;
name
	:NAMESTARTCHAR (namechar)*
	;
QNAME
	:	('a'..'z')+
	;
DIGITS
	:('0'..'9')+
	;
STRINGLITERAL
	:	('A'..'Z'| 'a'..'z'| '0'..'9')*
	;
S
	: (' '| '\r'|'\t'| '\n')+ {$channel=HIDDEN;}
	;
namechar
	:NAMESTARTCHAR | '-' | '.' | '0'..'9'
	;
NAMESTARTCHAR
	:':' | 'A'..'Z' | '_' | 'a'..'z'
	;
```

Appendix B: Manuscript of ADVCIT Paper

Jurnal Teknologi

Full Paper

A GPFCSP BASED FUZZY XQUERY INTERPRETER

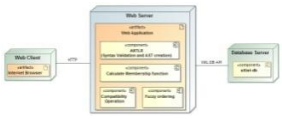
Pannipa Sae Ueng[a*], Srđan Škrbić[a], Supaporn Kansomkeat[b], Apirada Thadadech[b]

[a]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia
[b]Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand

Article history
Received
30 July 2015
Received in revised form
9 November 2015
Accepted
25 November 2015

*Corresponding author
pannipa@dmi.uns.ac.rs

Graphical abstract



Abstract

Nowadays XQuery has become the strongest standard for querying XML data. However, most of the real world information is in the form of imprecise, vague, ambiguous, uncertain and incomplete values. That is why there is a need for a flexible query language in which users can formulate queries that arise from their own criteria. In this paper, we propose an implementation of the Fuzzy XQuery - an extension of the XQuery query language based on the fuzzy set theory. In particular, we provide priority, threshold and fuzzy expressions for handling flexible queries. In addition, we have implemented an interpreter for this language by using the GPFCSP concept in Java and eXist-db environment.

Keywords: *Fuzzy XQuery; XQuery Interpreter; XML database; fuzzy set theory*

1.0 INTRODUCTION

XQuery language has been proposed as a standard for XML querying. It provides a feature called a FLWOR expression that supports the iteration and binding of variables to intermediate results. The FLWOR is an acronym: FOR, LET, WHERE, ORDER BY, RETURN, which is a powerful and important part of XQuery, similar in some aspects to the SQL query language in relational databases.

In real life, most information is imprecise, vague, ambiguous, uncertain or incomplete. Ideas related to improving query languages that can include such imprecise information in terms of the user's criteria is therefore natural. However, XQuery does not support the use of this kind of information by itself. In an effort to enrich it in a suitable manner, we have attempted to use the fuzzy set theory to provide a more flexible XQuery language, namely "Fuzzy XQuery". The Fuzzy XQuery is based on the standard XQuery v.1.0 with an added priority, threshold and fuzzy expressions. The interpreter for Fuzzy XQuery has been developed by using Java programming language and the eXist-db database. We can calculate the global constraint satisfaction degree of the result set with the concept of Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) [1] [2].

This paper is organized as follows. The next section contains the literature review. The third section presents the definition of the GPFCSP, compatibility operation and fuzzy ordering options. The architecture and implementation are shown in the fourth section. The fifth section presents an illustrative example and the last section is the conclusion.

2.0 RELATED WORKS

In this section we briefly review the main approaches of the flexible query techniques focusing on the application of fuzzy set theory.

Škrbić et al. proposed an extension of SQL with fuzzy capabilities called PFSQL (Prioritized Fuzzy Structured Query Language) [3]. A PFSQL interpreter was implemented using the priority fuzzy logic that is based on the concept of GPFCSP.

Many attempts for fuzzy querying in XML documents have been made in recent years. Campi et al. [4] presented FuzzyXPath, an attempt to enhance the flexibility of XPath. They introduced two

fuzzy constraints: CLOSE and SIMILAR applied to specific items within XML documents. Moreover, they also defined two flexible conditions for the flexible matching of path structures: BELOW and NEAR. Goncalves and Tineo [5] extended XQuery with the new xs:truth built-in data type to represent gradual truth degrees and xml:truth attribute of type xs:truth to handle the satisfaction degree in nodes of fuzzy XQuery expressions. Their language extension allowed users to declare fuzzy terms and used them in query expressions. Fredrick and Radhamani [6] illustrated their fuzzy XQuery techniques that allowed users to use linguistic terms based on the user-defined function. After that, in 2010 [7], they extended their earlier work by implementing the GUI tool with VB.net for the automatic generation of XQuery and fuzzy XQuery queries. In 2011 [8], they described the fuzzy XQuery process which used the arithmetic operations on fuzzy sets. Recently in 2012 [9], they defined fuzzy information in XML documents and the fuzzy domain integrity constraints through XML schemas for restricting invalid XML data into the XML database.

Panić et al. [10] implemented a similar approach as presented in this paper. They presented the fuzzy XML and fuzzy XQuery extension which used GPFCSP expressions, priority expressions and threshold expressions. The GPFCSP concept was used to calculate the membership degree in the same way as we did. In addition, they also developed a tool for working with XML, XSD and DTD documents, and fuzzy XQuery extension queries. However, the main difference between Panić's work and our work is that Panić's implementation used .NET framework, MATLAB and the Microsoft SQL Server database on a windows based application, whereas our approach used Java programming language to implement the new interpreter that was independent of MATLAB with eXist-db – native XML database on web based application.

3.0 BACKGROUND

3.1 Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP)

Skrbić et al. [1] [2] proposed the concept of GPFCSP for calculating the fuzzy membership degrees of PFSQL in fuzzy relational databases.

Theorem the following system (X, D, C^f, ρ, g, ∧, ∨, ¬, ◊) where

1. X = {x_i | i = 1, 2, …, n} is a set of variables,
2. D ={d_i | i=1, 2, …, n} is a set of domains. Every domain d_i is a set that contains possible values of variable $x_i \in X$,
3. C^f is a set of fuzzy constraints:

$$C^f=\{R_i^f \mid \mu_{R_i^f} : d_{i1} \times \ldots \times d_{ik_i} \rightarrow [0, 1], i = 1,\ldots,m, 1 \le k_i \le n\}$$

Where R_i^f denotes the set of constraint variables,

4. ρ: $C^f \rightarrow [0,\infty)$ is the priority of each constraint,
5. g:$[0,\infty)\times[0,1]\rightarrow[0,1]$ is the global satisfaction degree,
6. ∧ = T_L,
7. ∨ = S_L,
8. ¬ = 1-x,
9. ◊(x_i, c_i) = S_P (x_i, 1-ρ(c_i)), ρ(c_i) represents its priority,
10. v_X is a simultaneous valuation $v_X(x_1,\ldots,x_n)$, $x_i \in d_i$ of all variables in X

is a GPFCSP. The global satisfaction degree of a valuation v_X for a formula F is obtained in the following way:

$$\alpha_F(v_X) = F\{\Diamond (v_{x_i}, \frac{\rho(R^f)}{\rho_{\max}}) \mid R^f \in C^f \},$$

Where C^f is the set of constraints of formula F, $\rho_{max} = \max\{\rho(R^f), R^f \in C^f\}$.

In a similar way, we use the concept of GPFCSP to calculate the global satisfaction degree of Fuzzy XQuery because of the *where* clause in a FLWOR expression that contains a sequence of constrains connected with logical operators in the same way as in PFSQL.

3.2 Compatibility Operation

We can compare the fuzzy values in Fuzzy XQuery queries using standard notation fuzzyvalue1 = fuzzyvalue2. For example, $x/age = triangle(25,30,35). However, in order to allow the use of fuzzy values in Fuzzy XQuery queries, we need to calculate the compatibility of two fuzzy sets to measure to what extent one fuzzy set is a subset of some other fuzzy set.

Definition [2] Let A and B be two fuzzy sets over universe X. The measure of compatibility of the set A to the set B is defined as:

$$\text{Compatibility value } (C_{A,B}) = \frac{P(A \cap B)}{P(A)} \quad (1)$$

Where $P(A\cap B)$ is the area of intersection between two fuzzy sets and P(A) is the area of the fuzzy set A.

In our previous work, we implemented the modules for calculations of compatibility operations. There are three steps needed to calculate the compatibility in our approach [11]. Firstly, we define the algorithms to find the coordinates of the intersection area of two fuzzy sets. Secondly, the size of the shape of the intersection area is calculated as in equation (2).

$$\text{Area} = \left| \frac{(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + \ldots + (x_ny_1 - x_1y_n)}{2} \right| \quad (2)$$

Lastly, the compatibility value is calculated using equation (1).

3.3 Fuzzy Ordering

As mentioned before, we have defined fuzzy values in Fuzzy XQuery queries like fuzzyvalue1 = fuzzyvalue2. However, we can compare two fuzzy sets with the relational operators: >, >=, <, <= like fuzzyvalue1 > fuzzyvalue2. For example, $x/age>triangle(25,30,35).

In this case, we need to calculate the fuzzy ordering of two fuzzy sets. One usable definition of fuzzy ordering was proposed by Bodenhofer [12]. An ordering of fuzzy sets A and B is generalized as:

$$A \leq_I B \Leftrightarrow LTR(A) \supseteq LTR(B) \text{ and } RTL(A) \subseteq RTL(B) \quad (3)$$

LTR(A) stands for Left-to-Right closure which is the smallest fuzzy superset of A with a non-decreasing characteristic function, while RTL(A) stands for Right-to-Left closure which is the smallest fuzzy superset of A with a non-increasing characteristic function. We have proposed the algorithms and developed modules for fuzzy ordering calculations in [13].

4.0 IMPLEMENTATION

4.1 Designing the Fuzzy XQuery Grammar

We recall the extension of the XQuery language in EBNF (Extended Backus-Naur Form) from [14].

FLWORexpr	::= ForClause | LetClause WhereClause? OrderClause? ReturnClause
WhereClause	::= 'where' ExprSingle **(ThresholdExpr)**?
ExprSingle	::=OrExpr
ThresholdExpr	**:: 'threshold' DegreeLiteral**
OrExpr	::=AndExpr ("or" AndExpr)*
AndExpr	::=ComparisonExpr ("and" ComparisonExpr)*
ComparisonExpr	::=ValueExpr((GeneralComp)ValueExpr)
ValueExpr	::=pathexpr | **FuzzyExpr (PriorityExpr**)?
GeneralComp	::= '=' | '!= ' | '<' | '<=' | '>' | '>='
FuzzyExpr	**::'#' 'ling' '('QNAME')' '#'**
	| '#' 'tri' '('leftoffset','max','rightoffset')' '#'
	| '#' 'trap' '('leftoffset', 'leftmax', 'rightmax', 'rightoffset')' '#'
	| '#' 'interval' '('leftoffset','rightoffset')' '#'
	| '#' 'fs' '('type','leftoffset','rightoffset',)' '#'
PriorityExpr	**::="priority" DegreeLiteral**

Having this listing in mind, we conclude that our approach extends XQuery in the following points.

- Threshold Expression is an expression with the keyword *threshold* that removes results that have a membership degree to the result set that is less than the specified threshold value in a Fuzzy XQuery query. If there is no threshold expression, we assume that the value is 0.
- Priority Expression is an expression with the keyword *priority* that defines the level of influence of the corresponding constraints on the result. If the query does not specify the priority expression, the default value is 1.
- Fuzzy Expression is an expression that allows users to specify fuzzy numbers in XQuery queries. There are five types of fuzzy constants:
  - 'ling''('QNAME')' means a linguistic label with the name given by QNAME.
  - 'tri''('leftoffset','max','rightoffset')' means a Triangle fuzzy number with three arguments: left offset, maximum and right offset.
  - 'trap''('leftoffset','leftmax','rightmax', 'rightoffset')' means a Trapezoidal fuzzy number with four arguments: left offset, left maximum offset, right maximum offset and right offset.
  - 'interval''('leftoffset','rightoffset')' means an Interval fuzzy number with 2 arguments: left offset and right offset.
  - 'fs''('type','leftoffset','rightoffset')' means a Fuzzy Shoulder with 3 arguments: type of Fuzzy Shoulder (left shoulder or right shoulder), left offset and right offset.

4.2 System Architecture

The overall picture of the system's architecture is shown in Figure 1. There are two main parts:

1) eXist-db [15]: eXist-db is an open source software written in Java that is freely available in both source code and binary form. We have chosen the eXist-db database to store our XML documents because it provides for a pluggable module interface that allows for an extension modules to be easily developed in Java. These extension modules have full access to the eXist-db for XQuery query execution.

2) Interpreter: We implemented a parser for the Fuzzy XQuery grammar with ANTLR (ANother Tool for Language Recognition) version 3.4 [16]. ANTLR is the tool for the automatic generation of a lexical analyzer and a parser for a given EBNF grammar. We implemented an interpreter for the Fuzzy XQuery using Java in four main modules: *Transformer* transforms a Fuzzy XQuery to a standard XQuery, *CalculateMembershipFunction* uses the GPFCSP concept to calculate the membership degree of the results, *Compatibility Operation* calculates the compatibility operation of two fuzzy sets and *Fuzzy Ordering* calculates the ordering operation of two fuzzy sets. Moreover, we developed a web application GUI for the interpreter.

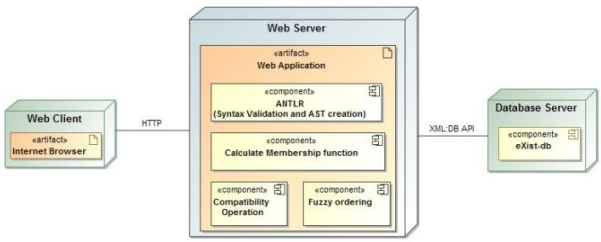


Figure 1 Architecture of the Fuzzy XQuery Interpreter

4.3 Fuzzy XQuery Execution

We implemented an interpreter that allowed for the execution of the Fuzzy XQuery queries defined above. The execution process is shown in Figure 2.

Let us explain in detail how we execute a Fuzzy XQuery. The system first checks the syntax of the Fuzzy XQuery following the given EBNF grammar. After that, if it is valid, the Fuzzy XQuery is transformed to a standard XQuery by parsing the Fuzzy XQuery,

creates an Abstract Syntax Tree (AST) and extracts the fuzzy part from it. Next, the system sends the standard XQuery into the database. When the database returns the result set, the system will interpret this result set again using the GPFCSP concept to calculate the membership degree of every element of the result set. Now we have the results that have a fuzzy membership degree in every element. Then, if the query has a threshold expression, the system will remove the tuples which have the fuzzy membership degree under the threshold value. Finally, we print the output to an XML file.

Now we describe how to calculate the fuzzy membership degrees in detail. After we have some result set that was obtained from a standard XQuery query, we calculate the fuzzy membership degree for every element of the result set. We walk the Fuzzy XQuery tree and find the WHERE node. Next, we traverse the *whereclause* subtree. If the current node represents a conjunction (AND) or disjunction (OR) node, we calculate the global constraint satisfaction degree (α) by calling the Łukasiewicz triangular norm (T_L) function or the Łukasiewicz triangular conorm (S_L) function, respectively. However, if the current node is an operator (=, !=, <, <=, >, >=), we walk its child node and check the type of the child node. If it is "/", we get the variable after this node. On the other hand, if it is a fuzzy constant (ling, tri, trap, interval or fs), we read the linguistic variable or offsets after this node. After that, we check the type of the operator. We calculate the membership degree by calling the compatibility operation module when an operator is an equality operator (=) or inequality operator (!=). However, if the operator is a relational operator (<, <=, >, >=), we call the fuzzy ordering module. Finally, if there is a child node with a priority expression, we aggregate the obtained value with a priority using the triangular product conorm (S_P).

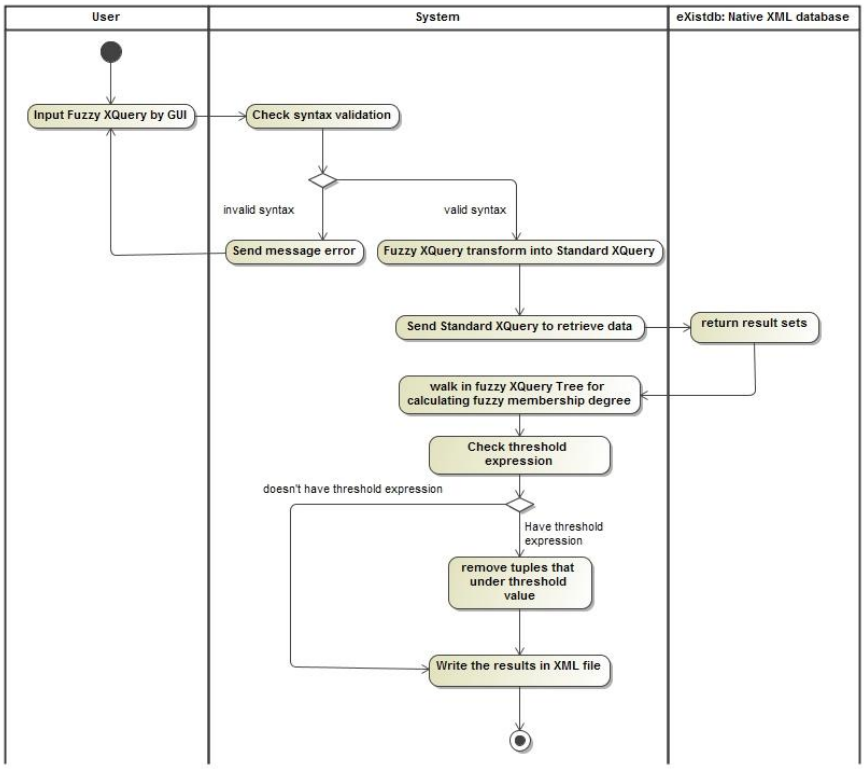


Figure 2 Activity diagram for executing a Fuzzy XQuery

5.0 ILLUSTRATIVE EXAMPLE

In this section, we illustrate the execution of the process of Fuzzy XQuery query with an example. Suppose that we have a Fuzzy XQuery query as in Listing 1 that retrieves the students who are of *young* age and their height is more than 150 cm with the priority 0.6 and 0.3, respectively. In addition, we define the threshold value equal to the 0.5 meaning that we want the results that have the global constraint satisfaction degree more than 0.5

Listing 1 An example of a Fuzzy XQuery query

```
for $x in document("student.xml") where $x/GPA >2.75 and
$x/age = #ling('young')# priority 0.6 and
$x/height > #tri(100,150,200)# priority 0.3
Threshold 0.5
return $x
```

Let us now describe how to calculate this Fuzzy XQuery. First of all, we transform the Fuzzy XQuery to a standard XQuery by removing the fuzzy expressions,

priority expressions and threshold expression as shown in Listing 2.

Listing 2 Transformation a Fuzzy XQuery to a standard XQuery query

```
for $x in document("student.xml")
where $x/GPA >2.75 return $x
```

Second, we get the result set after we send the standard XQuery to the database. Third, we send the results back to the interpreter to calculate the global constraint satisfaction degree by calling CalculateMembershipFunction. In this function, the system will remove the non-fuzzy conditions from the Fuzzy XQuery, which in this example is "$x/GPA >2.75", as in Listing 3.

Listing 3 The Fuzzy XQuery after removing the non-fuzzy node

```
for $x in document("student.xml")
where $x/age = #ling('young') priority 0.6 and
$x/height > #tri(100,150,200)# priority 0.3
Threshold 0.5 return $x
```

We use the concept of GPFCSP (as in the preceding section) to calculate the global constraint satisfaction degree for all the result set in step two by using the equation (4).

$$\alpha = T_L(S_P(\mu_{R_1^f}(v), 1-\rho(R_1^f)), S_P(\mu_{R_2^f}(v), 1-\rho(R_2^f))) \quad (4)$$

In the equation (4), R_i^f is the fuzzy constraint i and $\mu_{R_i^f}$ is the satisfaction degree of constraint R_i^f. The priority of each constraint is represented by the function $\rho(R_i^f)$. The greater value of $\rho(R_i^f)$ means that the constraint R_i^f is more important. In this example, the constraint R_1^f: *age* is more important than the constraint R_2^f: *height* because the priority value of the constraint *age* is 0.6 but the priority value of the constraint *height* is 0.3. It is noticeable that we use the T_L because of the conjunction AND in this Fuzzy XQuery. The S_P is used to aggregate with priority.

Let us assume that we have the student data in the XML file as in Listing 4 and the result set from the standard XQuery is shown in Listing 5. It is noticeable that Ana's GPA is not greater than 2.75. Consequently, the result in Listing 5 does not show Ana's record.

We calculate the constraint satisfaction degree ($\mu_{R_i^f}$) for every constraint and every student as in Table I. In the case of the first constraint *age*, these degrees are obtained directly as the values of the corresponding membership functions of the *young* linguistic fuzzy variable at the given point of the age data. Suppose that we define the linguistic value of *young* in an XML document whose membership function have the left fuzzy shoulder which can be seen in Figure 3. However, with the second constraint *height*, we calculate $\mu_{R_2^f}$ by using the fuzzy ordering modules since the type of the fuzzy constant is *tri* and the operator is >. If we substitute $\mu(R_i^f)$ and $\rho(R_i^f)$ for the first student (John) into the equation (4), we obtain the following:

$$\alpha_{John} = T_L(S_P(0,1-0.6), S_P(0.5,1-0.3)) \quad (5)$$

Therefore, we obtain the global constraint satisfaction degree of John as follows:

$$\alpha_{John} = T_L(S_P(0,0.4), S_P(0.5,0.7)) = T_L(0.4,0.85) = 0.25 \quad (6)$$

Listing 4 The snippet of student data

```
<?xml version= "1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>John</name>
        <GPA>3.5</name>
        <age>25</age>
        <height>170</height>
    </student>
    <student>
        <name>Peter</name>
        <GPA>3.0</name>
        <age>21</age>
        <height>165</height>
    </student>
    <student>
        <name>Ana</name>
        <GPA>2.5</name>
        <age>22</age>
        <height>180</height>
    </student>
    <student>
        <name>Alex </name>
        <GPA>2.8</name>
        <age>20</age>
        <height>tri(150,200,250)</height>
    </student>
</students>
```

Listing 5 The result set from standard XQuery in Listing 2

```
<?xml version= "1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>John</name>
        <GPA>3.5</name>
        <age>25</age>
        <height>170</height>
    </student>
    <student>
        <name>Peter</name>
        <GPA>3.0</name>
        <age>21</age>
        <height>165</height>
    </student>
    <student>
        <name>Alex </name>
        <GPA>2.8</name>
        <age>20</age>
        <height>tri(150,200,250)</height>
    </student>
</students>
```

The other students are calculated in the same way and are given in Table II.

Table 1 The constraint satisfaction degrees of every constraint and every student

Name	$\mu_{R_1^f}$	$\mu_{R_2^f}$
John	0	0.5
Peter	0.8	0.5
Alex	1	1

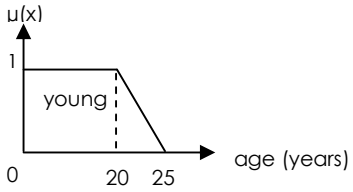


Figure 3 Membership function of *young*

Table 2 The global constraint satisfaction degrees (α) of every student

Name	α
John	0.25
Peter	0.73
Alex	1

Finally, because of the threshold value, the system will print the results which have the global constraint satisfaction degree more than 0.5 as shown in Listing 6.

Listing 6 The final result set

```
<?xml version= "1.0" encoding="UTF-8"?>
<results>
        <student>
                <name>Peter</name>
                <alpha>0.73</alpha>
        </student>
        <student>
                <name>Alex</name>
                <alpha>1.0</alpha>
        </student>
</results>
```

6.0 CONCLUSION

In this paper, we present an approach that uses the fuzzy set theory that can manage the imprecise, vague, ambiguous, uncertain or incomplete data with XML technology. We have proposed extensions for the XQuery query language in order to handle flexible fuzzy queries that provide *priority*, *threshold* and *fuzzy* expressions. Furthermore, we implement an interpreter for this language and web GUI using Java programming language and eXist-db. The GPFCSP concept is used to calculate the global constraint satisfaction degrees. In the future, we plan to test the performance of our application with different case studies and develop a more modern web application using AngularJS.

Acknowledgement

This work was supported by the budget revenue from Prince of Songkla University and Faculty of Science, Prince of Songkla University, Thailand, through the project no. SCI570329S: A Fuzzy XML Database System and partially supported by the Ministry of Education and Science of the Republic of Serbia, through the project no. 174023: Intelligent techniques and their integration into the wide-spectrum decision support.

References

[1] Škrbić, S., Racković, M. and Takaši, A. 2013. Prioritized fuzzy logic based information processing in relational databases. *Knowledge Based Systems*. 38:62–73.
[2] Škrbić, S., Racković, M. 2013. *FUZZY DATABASES*. Novi Sad, Serbia: Faculty of Sciences.
[3] Škrbić, S. and Racković, M. 2009. Pfsql: a fuzzy sql language with priorities. The 4th International *Conference on Engineering Technologies (ICET), 2009* Novi Sad, Serbia. 28-30 April 2009. 119–125.
[4] Campi, A., Damiani, E., Guinea, S., Marrara, S., Pasi, G., and Spoletini, P. 2009. A fuzzy extension of the XPath query language. *Journal of Intelligent Information Systems*. 33:285–305.
[5] Goncalves, M. and Tineo, L. 2010. Fuzzy XQuery. In *Soft Computing in XML Data Management*. Series Studies in Fuzziness and Soft Computing. Ma, Z. and Yan, L. (ed.) Springer Berlin/Heidelberg. 255:133–163.
[6] Fredrick, E. T. and Radhamani, G. 2009. Fuzzy logic based XQuery operations for native XML database systems. *International Journal Database Theory and Application*. 2:13–20.
[7] Fredrick, E. T. and Radhamani, G. 2010. A GUI based tool for generating XQuery and fuzzy XQuery. *International Journal of Computer Applications Database Theory and Application*. 1:54–58.
[8] Fredrick, E. T. and Radhamani, G. 2011. Information retrieval using XQuery processing techniques. *International Journal of Database Management Systems (IJDMS)*. 3:50–58. Feb, 2011.
[9] Fredrick, E. T. and Radhamani, G. 2012. Fuzzy integrity constraints for native xml database. *International Journal of Computer Science (IJCSI)*. 9:50–58. Mar, 2012.
[10] Panić, G., Škrbić, S. and Racković, M. 2014. Fuzzy xml and prioritized fuzzy xquery with implementation. *Journal of Intelligent and Fuzzy Systems*. 26:303–316.
[11] Sukpisit, S., Kansomkeat, S., Thadadech, A., Ueng, P. S. and Škrbić, S. 2015. Polygon intersection based algorithm for fuzzy set compatibility calculations. *2015 International Conference on Information Technology (ICIT), 2015*. Singapore. 2-3 Feb. 2015. 241–248.
[12] Bodenhofer, U. 2008. Orderings of fuzzy sets based on fuzzy orderings part i: The basic approach. *Mathware Soft Computing*. 15:201–218.
[13] Kansomkeat, S., Sukpisit, S., Thadadech, A., Ueng, P. S. and Škrbić, S. 2015. Fuzzy ordering implementation applied in fuzzy XQuery. *5th International Conference on Information Society and Technology (ICIST), 2015*. Kopaonik, Serbia. 8-11 March 2015. 443–448.
[14] Thadadech, A., Kansomkeat, S., Vonghirandecha, P. and Škrbić, S. 2015. A Fuzzy XML Database System. Final report of collaborative research. Prince of Songkla University, Thailand.
[15] eXistdb project. 2014. eXistdb. [Online]. From: http://existdb.org/exist/apps/homepage/index.html. [Accessed on 3 July 2015].
[16] Parr, T. 2012. ANTLR v3. [Online]. From: http://www.antlr3.org. [Accessed on 10 June 2015]

Appendix C: Paper of IJMLC Journal

Polygon Intersection Based Algorithm for Fuzzy Set Compatibility Calculations

Sukgamon Sukpisit, Supaporn Kansomkeat, Pannipa Sae Ueng, Apirada Thadadech, and Srđan Škrbić

***Abstract*—PFSQL is an extension of the SQL language that allows usage of fuzzy logic in SQL queries. In query statements, variables can take both fuzzy and non-fuzzy values. Normally, different types of values cannot be compared directly. Therefore, it is necessary to implement fuzzy compatibility calculation to solve this problem. This paper proposes a method of fuzzy compatibility calculation implementation that determines compatibility degree of two fuzzy sets. The compatibility value is calculated using polygon intersection algorithm. To prove the correctness of the proposed method, the application has been developed and tested with 360 compatibility cases of different randomly generated fuzzy values. The experimental results show that our algorithms can handle various types of intersections between any two fuzzy sets.**

***Index Terms*—Compatibility, fuzzy database, fuzzy query, PFSQL.**

I. Introduction

In the real world applications, some information might be vague, ambiguous, uncertain, imprecise or incomplete. Fuzzy logic has become a successful approach to handle this kind of information [1]. At the same time, methods of incorporating fuzziness into relational databases, such as fuzzy data models that are introduced by Ma *et al*. [2] and Vucetic *et al*. [3], are studied. In 2012, Škrbić and Racković introduced PFSQL (Prioritized Fuzzy Structured Query Language) that represents a set of extensions to SQL using priority fuzzy logic, together with a new fuzzy relational data model based on fuzzy extensions of the relational model [1].

PFSQL allows fuzzy logic concepts to be used in queries. Variables in query statements can be assigned both fuzzy and crisp values [4]. For example, a.wealth = triangle (13, 18, 20). Normally, a non-fuzzy value (a.wealth) and a fuzzy value (triangle (13, 18, 20)) cannot be compared directly because they are of different type. To solve this problem, the fuzzy compatibility calculations must be used. In this paper, we propose a method of implementation of fuzzy compatibility calculations between fuzzy sets. Our algorithm is capable of calculating intersection of every pair of the following types: triangular fuzzy number, trapezoidal fuzzy number, intervals, fuzzy shoulders and crisp values. This algorithm may then be used for wide spectrum of problems, but our interest is to use it for the implementation of different types of fuzzy queries. For example, it can be applied to the implementation of an interpreter for the fuzzy XQuery language proposed by Ueng and Škrbić in [5].

This paper is organized as follows. In the next section, we introduce the algorithms that we propose for compatibility calculations. Our implementation and testing results are presented in Sections III and IV, respectively. Section V is the conclusion.

Manuscript received May 20, 2015; revised January 18, 2016. Authors are partially supported by Ministry of Education and Science of the Republic of Serbia, through project no. ON 174023: Intelligent techniques and their integration into wide-spectrum decision support.

S. Sukpisit, S. Kansomkeat, and A. Thadadech are with the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand (e-mail: sukgamon.s@psu.ac.th, supaporn.k@psu.ac.th, apirada.t@psu.ac.th).

P. S. Ueng and S. Škrbić are with the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia (e-mail: pannipa@dmi.uns.ac.rs, srdjan.skrbic@dmi.uns.ac.rs).

II. Compatibility Calculation

Our research focuses on five fuzzy types: triangular fuzzy numbers, trapezoidal fuzzy numbers, fuzzy shoulders, intervals and crisp values. In this section we describe the algorithm capable of determining the compatibility degree of two fuzzy sets of those types.

The compatibility calculation process is separated into three steps. First, the intersection area of two fuzzy sets is determined. Second, the size of the shape of the intersection area is calculated. Finally, a compatibility value is obtained using the compatibility equation.

A. Determining Intersection Area

An intersection area of two fuzzy sets is determined in 2-dimensions: vertical (x) and horizontal (y). We can assume that the shape of any characteristic function is a polygon. Each edge of a polygon can be transformed into linear equation $(y = mx + c)$ and used for calculations in that form. For example, a fuzzy triangle shape has 3 coordinates: (*LeftOffset*, 0), (*Maximum*, 1) and (*RightOfsset*, 0). The bottom edge ($y = 0$) is not used for compatibility computation, so a fuzzy triangle have two edge-equations, *LeftEdge* and *RightEdge*. Table I shows coordinates and edge-equations of all characteristic functions used in this paper. A fuzzy trapezoidal shape has two edge-equations same as triangle and one additional edge-equation called *CenterEdge* which is simple – $y = 1$.

There are two types of fuzzy shoulder shapes – ascending or right shoulder and descending or left shoulder. There are two edge-equations in both types: *LeftEdge* and *RightEdge*. One edge-equation of them is constant depending on its type. A fuzzy interval is a line graph that starts from (*LeftMax*, 1) and ends at (*RightMax*, 1). The area below that line graph gives a rectangular shape that can be calculated easily but it is more complex to determine a common area with other shapes.

The main activity in determining intersection area step is the coordinate and edge-equations specification of the intersection area. The coordinates are transformed into

doi: 10.18178/ijmlc.2016.6.1.567

objects. Each object provides *x* and *y* coordination called *Coordinate*. All objects are clockwise or counter clockwise added to a coordination table. The sequence of adding is important because non-consecutive adding may lead to incorrect results in the polygonal area calculation step.

This paper considers five types of fuzzy sets. For the sake of brevity, only the simplest case, triangle and triangle, is demonstrated in this paper. In this case, we only use left and right edges to find coordinates of the intersection area. Listing 1 shows the algorithm that calculates coordinates of the intersection area.

TABLE I: COORDINATES AND EDGE-EQUATION OF ALL CHARACTERISTIC FUNCTIONS

Characteristic functions	Coordinates	Edge equations
Triangle		
	(LeftOffset, 0)	LeftEdge, RightEdge
	(Maximum, 1)	
	(RightOffset, 0)	
Trapezoidal		
	(LeftOffset, 0)	LeftEdge, CenterEdge, RightEdge
	(LeftMaximum, 1)	
	(RightMaximum, 1)	
	(RightOffset, 0)	
Right Shoulder		
	(ZeroPoint, 0)	LeftEdge, RightEdge
	(Maximum, 1)	
	(∞, 1)	
	(∞, 0)	
Left Shoulder		
	(0, 0)	LeftEdge, RightEdge
	(0, 1)	
	(Maximum, 1)	
	(ZeroPoint, 0)	
Interval		
	(LeftMaximum, 0)	
	(LeftMaximum, 1)	
	(RightMaximum, 1)	
	(RightMaximum, 0)	
Crisp value		
	(*X*, *Y*)	

LISTING. 1: ALGORITHM FOR DETERMINING COORDINATES OF THE INTERSECTION AREA BETWEEN TWO FUZZY TRIANGLES

Algorithm GetCoordinates (FuzzyTriangle A, FuzzyTriangle B).

01. Compare *LeftOffset* and *RightOffset* of 2 fuzzy sets.
02. If there is intersection area
03. Store a coordinate ($LeftOffset_{max}$, 0).
04. Find a coordinate of interception of $LeftEdge_A$ and $LeftEdge_B$.
05. Find a coordinate of interception of $LeftEdge_A$ and $RightEdge_B$.
06. Find a coordinate of interception of $RightEdge_A$ and $LeftEdge_B$.
07. Find a coordinate of interception of $RightEdge_A$ and $RightEdge_B$.
08. Store coordinate ($RightOffset_{min}$, 0).
09. End if
10. End

The algorithm starts from checking intersection area of two fuzzy sets by comparing *LeftOffset* and *RightOffset* for the two. If there is no intersection area, the algorithm ends. If there is an intersection area, the first coordinate of it is ($LeftOffset_{max}$, 0). The $LeftOffset_{max}$ can be obtained by calculating the maximum value of $LeftOffset_A$ and $LeftOffset_B$. This coordinate is stored in the coordination table. In order to find and store coordinates in clockwise direction, we start by comparing $LeftEdge_A$ with $LeftEdge_B$. If these two edges overlap, the coordinates of the overlapping point is stored in the second row of the coordination table. To find and store the next three coordinates, method proceeds in the same manner. The last coordinate is obtained by calculating the minimum value of $RightOffset_A$ and $RightOffset_B$.

B. Calculating Intersection Area

In our research, the cyclic polygon calculation proposed in [6] is used to calculate the intersection area. This method uses coordinates of a polygon for the area calculations. The area is calculated by the following equation:

$$\text{Area} = \left|\frac{(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + \cdots + (x_ny_1 - x_1y_n)}{2}\right| \quad (1)$$

To demonstrate the process, we describe compatibility calculation for two fuzzy sets: *triangleA* (12, 15, 18) and *triangleB* (14, 16, 17). The three attributes of a triangular fuzzy number are *LeftOffset*, *Maximum* and *RightOffset* respectively. Fig. 1 shows the fuzzy sets, *triangleA* and *triangleB*. There are four coordinates of the intersection of these two fuzzy sets. Fig. 2 shows coordinates of *triangleA* and *triangleB*, and their coordination table.

When you submit your final version, after your paper has been accepted, prepare it in two-column format, including figures and tables.

$$\left|\frac{((14\times0.8)-(15.6\times0))+((15.6\times0.5)-(16.5\times0.8))+((16.5\times0)-(17\times0.5))+((17\times0)-(14\times0))}{2}\right| = 1.35$$

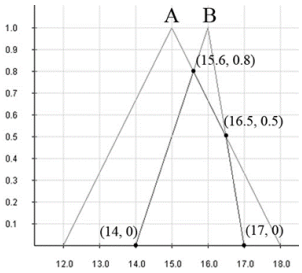


Fig. 1. *TriangleA* and *triangleB*.

Attr.	X	Y
LeftOffset	12	0
Maximum	15	1
RightOffset	18	0

triangleA

Attr.	X	Y
LeftOffset	14	0
Maximum	16	1
RightOffset	17	0

triangleB

X	Y
14	0
15.6	0.8
16.5	0.5
17	0

Coordination Table

Fig. 2. Coordinates of *triangleA* and *triangleB*, and their coordination table.

C. Calculating Compatibility Value

To obtain compatibility value, the compatibility equation [4] will be applied. The equation of compatibility of the fuzzy set A to the fuzzy set B is given below.

$$C = \frac{P(A \cap B)}{P(A)} \quad (2)$$

$P(A \cap B)$ is the area of intersection between the two fuzzy sets and $P(A)$ is the area of the source fuzzy set *A*.

Compatibility value is a number that varies from 0 to 1. Zero means incompatible, and one means fully compatible.

As stated before, we focus on five types of fuzzy sets. Each fuzzy set has a different shape of the characteristic function. An area of each shape can be obtained by mathematical methods. For example, the area of triangular fuzzy set can be calculated by $\left|\frac{\text{width}\times\text{height}}{2}\right|$.

In Fig. 1, the area of the intersection is 1.35. The area of the source fuzzy set (*triangleA*) is 3. Therefore, the compatibility value for these two fuzzy sets equals 0.45.

III. Implementation

To support our ideas, we developed the application that provides graphical user interface for calculation of the intersection area of two fuzzy sets and determining coordinates of the intersection area. Compatibility value is then obtained using this data. The application has two functions: manual testing and random testing. The manual testing function is used for a single test. User can identify two fuzzy sets and then this function will return the compatibility value. Fig. 3 shows the user interface for the manual testing function. The random testing function generates cases randomly. In this case, user can indicate type of fuzzy sets, boundary values and the number of generated cases. The boundary values include minimum value and maximum value. Fig. 4 shows the user interface of the random testing function.

Fig. 3. User interface of the manual compatibility testing application.

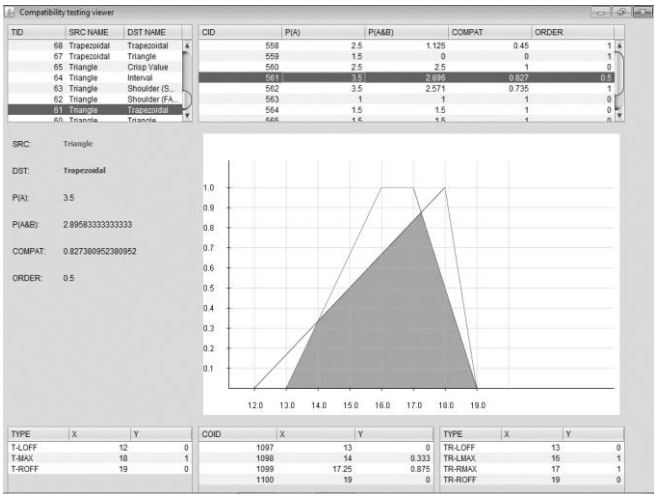


Fig. 4. User interface of the random compatibility testing application.

In our application, when the compatibility calculation processes are finished, the image of fuzzy sets and their intersection area will appear on the screen.

This application was developed on Java platform with the use of PostgreSQL to store fuzzy set attributes and cases of the random testing function.

IV. Testing Results

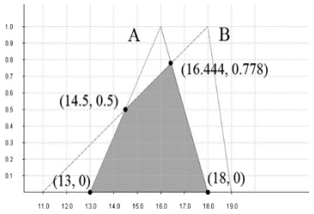
To prove the correctness of our algorithms, some compatibility cases are generated. To reduce the collecting bias, fuzzy sets are generated randomly using the described random testing function of our application. For thorough testing, each fuzzy type was compared with other types. For example, fuzzy triangle is compared with other four types including itself. There are two types of fuzzy shoulders, therefore there are six comparison pairs. Each pair has ten cases of compatibility testing. Totally, there are 360 compatibility cases in our experiment.

TABLE II: A Comparisons between Fuzzy Triangle and Other Types

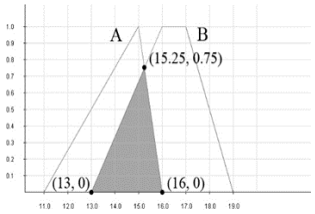
No.	Type of fuzzy A		P(A)	Type of fuzzy B		$P(A \cap B)$	C
	Attributes	Val.		Attributes	Val.		
1	Fuzzy triangle		2.5	Fuzzy triangle		2.2222	0.8889
	$LeftOffset_A$	13		$LeftOffset_B$	11		
	$Maximum_A$	16		$Maximum_B$	18		
	$RightOffset_A$	18		$RightOffset_B$	19		
2	Fuzzy triangle		2.5	Fuzzy trapezoidal		1.125	0.45
	$LeftOffset_A$	11		$LeftOffset_B$	13		
	$Maximum_A$	15		$LeftMax_B$	16		
	$RightOffset_A$	16		$RightMax_B$	17		
				$RightOffset_B$	19		
3	Fuzzy triangle		2.5	Fuzzy shoulder (FC)		1.553	0.6212
	$LeftOffset_A$	12		$ZeroPoint_B$	11		
	$Maximum_A$	15		$Maximum_B$	20		
	$RightOffset_A$	17					
4	Fuzzy triangle		2	Fuzzy shoulder (SB)		0.25	0.125
	$LeftOffset_A$	13		$Maximum_B$	10		
	$Maximum_A$	16		$ZeroPoint_B$	15		
	$RightOffset_A$	17					
5	Fuzzy triangle		2.5	Fuzzy interval		0.5	0.2
	$LeftOffset_A$	12		$LeftMax_B$	11		
	$Maximum_A$	16		$RightMax_B$	14		
	$RightOffset_A$	17					
6	Fuzzy triangle		3.5	Crisp value			0.3333
	$LeftOffset_A$	10		X_{crisp}	12		
	$Maximum_A$	16		Y_{crisp}	0.78		
	$RightOffset_A$	17					
7	Fuzz triangle		1.5	Fuzzy trapezoidal		0	0
	$LeftOffset_A$	10		$LeftOffset_B$	14		
	$Maximum_A$	11		$LeftMax_B$	17		
	$RightOffset_A$	13		$RightMax_B$	18		
				$RightOffset_B$	19		

If the range in the random testing function is too long, fuzzy sets can be positioned too far from each other and have no compatibility. To overcome this problem, the minimum and maximum values of boundaries are set to be 10 and 20, respectively.

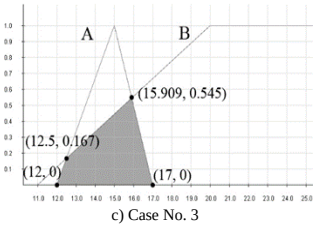
Table II shows some cases of comparisons between fuzzy triangle and other types. The intersection areas of the cases 1 to 5 have 4, 3, 4, 3 and 3 coordinates, respectively. The last two cases have no intersection area. The shapes and coordinates of intersection area of each case are shown in Fig. 5.



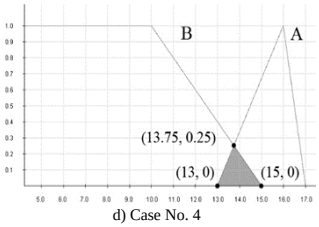
a) Case No. 1



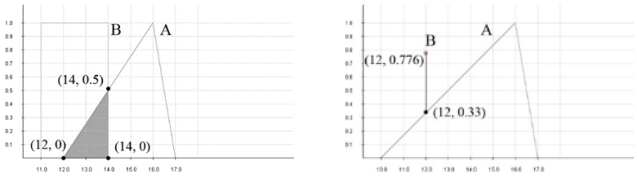
b) Case No. 2



c) Case No. 3



d) Case No. 4



e) Case No. 5 f) Case No. 6

Fig. 5. User interface of the random compatibility testing application.

V. Conclusion

This paper proposes the algorithm for fuzzy compatibility calculation of two fuzzy sets. The compatibility measure is used to compare two fuzzy sets. First we introduced the algorithms able to determine the intersection area between two fuzzy sets. After that, the compatibility calculation processes is explained and illustrated. Within the paper, the application that provides GUI for compatibility calculations is developed. The testing results are generated randomly by this application. In these results, various shapes of intersection areas are recognized correctly by our implementation. In this way we illustrated the power of the proposed algorithms to handle various types of intersections between any two fuzzy sets of the five fuzzy membership function types that we described.

It is our intent to use the proposed algorithms for compatibility calculations inside the interpreter for the fuzzy logic enriched XQuery language.

In the future, we plan to develop and implement the algorithms capable of calculating fuzzy ordering. Fuzzy ordering is important operation for queries that contain relational operators, as well as for those that contain aggregate functions like MIN, MAX and SUM.

References

[1] S. Škrbić, M. Racković, and A. Takači, "Prioritized fuzzy logic based information processing in relational databases," *Knowledge-Based Systems*, vol. 38, pp. 62-73, 2013.

[2] Z. M. Ma, F. Zhang, and L. Yan, "Fuzzy information modeling in UML class diagram and relational database models," *Applied Soft Computing*, vol. 11, pp. 4236-4245, 2011.

[3] M. Vucetic, M. Hudec, and M. Vujoševic, "A new method for computing fuzzy functional dependencies in relational database systems," *Expert Systems with Applications*, vol. 40, pp. 2738-2745 ,2013.

[4] S. Škrbić and M. Racković, *Fuzzy Databases*, Faculty of Sciences, University of Novi Sad, Novi Sad, 2013.

[5] P. S. Ueng and S. Škrbić, "Implementing XQuery fuzzy extensions using a native XML database," in *Proc. 13th IEEE International Symposium on Computational Intelligence and Informatics*, 2012, pp. 305-309.

[6] I. Pak, "The area of cyclic polygons: Recent progress on Robbins' conjectures," *Advances in Applied Mathematics*, vol. 34, pp. 690-696, 2005.

Sukgamon Sukpist is a master student at the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand. He received his B.Sc. degree in information and communication technology from Prince of Songkla University, Songkhla, Thailand in 2010. His research interest is fuzzy XQuery. He is a software developer at the Computer Center, Prince of Songkla University since 2010.

Supaporn Kansomkeat is an assistant professor at the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand. Her current research interests include software testing, test process improvement and fuzzy XQuery. She received a PhD in computer engineering from Chulalongkorn University in 2007. She was a publicity co-chair for the International Conference on Asia-Pacific Software Engineering Conference. She is a general secretariat for the 12th International Joint Conference on Computer Science and Software Engineering.

Pannipa Sae Ueng is a Ph.D. student at the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia. She received her B.Sc. degree in computer science from the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand and M.Sc. degree in computer science from the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand. She has worked as a lecturer at the Department of Computer Science, Faculty of Science, Prince of Songkla University since 2007. Her research interests are fuzzy database and fuzzy XQuery.

Apirada Thadadech is an assistant professor at the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand. She received her M.Sc. degree in computer science from University of Philippines Losbanos, Philippines in 1990. In 2013, she was elected as a head of the Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand. Her research interest is software engineering.

Srđan Škrbić is an associate professor at the Department of Mathermatics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia. He received his B.Sc. degree in computer science in 2001. M.Sc. in computer science in 2004 and Ph.D. in computer science in 2009 at the Faculty of Sciences, University of Novi Sad. In 2006, he received the "Mileva Maric – Einstein" prize for his M.Sc. thesis. He was elected as a head of the Chair for information technologies and systems in 2011. Recently he has focused on reseach topics in high performance and parallel scientific computing. He coauthored more than 50 research papers.

Appendix D: Proceeding of ICIST2015

Fuzzy Ordering Implementation Applied in Fuzzy XQuery

Supaporn Kansomkeat*, Sukgamon Sukpisit*, Apirada Thadadech*,
Pannipa Sae Ueng** and Srdjan Skrbic**
* Prince of Songkla University, Department of Computer Science, Songkhla, Thailand
** University of Novi Sad, Department of Mathematics and Informatics, Novi Sad, Serbia
supaporn.k@psu.ac.th, sukgamon.s@psu.ac.th, apirada.t@psu.ac.th,
pannipa@dmi.uns.ac.rs, srdjan.skrbic@dmi.uns.ac.rs

***Abstract* — Fuzzy XQuery is the extension of standard XQuery language that allows fuzzy values in the query condition statements. Relational operators are not only required and possible in crisp value cases but also for fuzzy values. When relational operators are included in the query, it is necessary to provide means for comparison between fuzzy sets. These fuzzy relational operators are typically used in two fuzzy sets comparison case, but can also be used with some aggregate functions like MIN, MAX, and SUM. The aim of this paper is to present the algorithms for the implementation of fuzzy relational operators. Our algorithms compare the horizontal positions of two fuzzy sets and calculate the ordering value based on partial fuzzy ordering proposed by Bodenhofer. Moreover, we developed a GUI application and evaluated our approach with 360 fuzzy ordering cases. The experimental results show that our algorithms are capable of calculating fuzzy ordering values with various types of fuzzy values correctly.**

I. INTRODUCTION

Recently, fuzzy extensions are proposed to handle vague, ambiguous, uncertain, imprecise or incomplete information. Campi et al. [1] introduced fuzzy extensions to XPath named FuzzyXPath that used to query XML data based on the fuzzy set theory. Fredrick and Radhamani [2] introduced fuzzy XQuery to retrieve data from native XML database. Skrbic et al. [3] introduced PFSQL (Prioritized Fuzzy Structured Query Language), which is an extension of SQL (Structured Query Language). PFSQL uses the prioritized fuzzy logic to retrieve data from a fuzzy relational database. In 2012, Ueng and Skrbic [4] proposed fuzzy extensions to standard XQuery. Their query system retrieves data from native XML database based on prioritized fuzzy logic. In 2014, they implemented an interpreter for fuzzy XQuery in their project called FXI (Fuzzy XQuery Interpreter). Users can query data with priority and threshold keywords in the condition statement and define fuzzy values used as search conditions in the query.

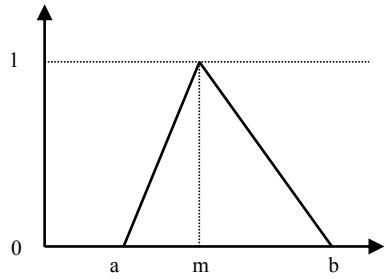
Including fuzzy relational operators in FXI is a very promising idea. In this way, fuzzy XQuery queries would be able to provide flexible comparisons between fuzzy sets that represent vague data. Relational operators on fuzzy sets are binary operators, which are able to compare two fuzzy sets: $<, \leq, \geq$ and $>$. Furthermore, fuzzy relational operators can be used with some aggregate functions like MIN, MAX, and SUM. In this paper, we propose a method to calculate fuzzy relational operations between two fuzzy sets and give its implementation. The proposed method is general and may be used with different types of problems. For example, it can be applied to fuzzy XQuery or PFSQL.

This paper is organized as follows. In the next section, we introduce algorithms for fuzzy relational operator calculations. Our implementation and testing results are presented in Sections 3 and 4, respectively. Section 5 is the conclusion.

II. FUZZY ORDERING CALCULATIONS

A. Membership functions

There are five different types of fuzzy membership functions used in [4]: triangle fuzzy number, trapezoidal fuzzy number, interval, fuzzy shoulder and crisp value. Figure 1 shows the shape of a fuzzy triangle membership function.



$$\mu_A(x) = \begin{cases} 0, & \forall x, x \leq a \wedge x \geq b \\ \dfrac{x-a}{m-a}, & \forall x \in (a, m] \\ \dfrac{b-x}{b-m}, & \forall x \in (m, b) \end{cases}$$

Figure 1 Fuzzy triangle number and its membership function

Definition 1 A fuzzy set A over universe X is determined by its characteristic (membership) function [5],

$$\mu_A: x \to [0,1],$$

where, for every $x \in X, \mu_A(x)$ is interpreted as membership degree of element x to fuzzy set A. Value $\mu_A(x) = 0$ denotes that element x does not belong to the set A, while $\mu_A(x) = 1$ denotes that element x belongs to the set A. Universe X is almost always the set of real numbers.

Definition 2 The set $x \in X \mid \mu_A > 0$ is called the support of A (*supp(A)*) and the set $\{x \in X \mid \mu_A = 1\}$ is called its kernel (*ker(A)*) [5]

B. Fuzzy ordering calculation

In 2008, fuzzy orderings were proposed by Bodenhofer in [6]. Here we recall some basic definitions used in our research; for a more extensive description see [6].

Definition 3 Consider a fuzzy equivalence relation, *T*-equivalence $E: X^2 \to [0,1]$ and a direct fuzzification, *T-E*-ordering $L: X^2 \to [0,1]$. Then, for given fuzzy set $A \in \mathcal{F}(X)$, where $\mathcal{F}(X)$ is a fuzzy superset of *X*. The fuzzy sets 'at least *A*' and 'at most *A*' (with respect to *L*), abbreviated ATL(*A*) and ATM(*A*), respectively, are defined as follow (for all $x \in X$):

$$ATL(A)(x) = \{T(A(y), L(y,x)) \mid y \in X\} \quad (1)$$

$$ATM(A)(x) = \{T(A(y), L(x,y)) \mid y \in X\} \quad (2)$$

ATL(*A*) is the smallest fuzzy superset of *A* that has a non-decreasing membership function with respect to *L*, while ATM(*A*) is the smallest fuzzy superset of *A* that has a non-increasing membership function with respect to *L*.

When *L* is a crisp ordering, the notations LTR(*A*) and RTL(*A*) are used instead of ATL(*A*) and ATM(*A*), respectively. LTR(*A*) stands for left-to-right closure and RTL(*A*) stands for right-to-left closure. The operator $\preccurlyeq$ is referred to crisp ordering.

$$LTR(A)(x) = \{A(y) \mid y \in X \wedge y \preccurlyeq x\} \quad (3)$$

$$RTL(A)(x) = \{A(y) \mid y \in X \wedge x \preccurlyeq y\} \quad (4)$$

First we describe a well-known ordering procedure for real intervals.

$$[a,b] \leq_I [c,d] \Leftrightarrow a \leq c \wedge b \leq \mathrm{d} \quad (5)$$

Equation (5) states that the only case that yields "true" or 1 value is $a \leq c$ and $b \leq d$. The inequality $a \leq c$ means that there are no elements of set [c, d] that are below the entire interval [a, b] and the inequality $b \leq d$ means that there are no elements of [a, b] that are completely above [c, d]. Equation (5) can be generalized to arbitrary crisp subsets of an ordered set $(x, \preccurlyeq)$ as follow:

$$M \preccurlyeq_I N \Leftrightarrow \big((\forall x \in N)(\exists y \in M) y \preccurlyeq x\big) \wedge \big((\forall x \in M)(\exists y \in N) x \preccurlyeq y\big) \quad (6)$$

By using the operators LTR and RTL, and considering a crisp ordering $\preccurlyeq$ on *X*, the following equivalences that hold for all $M, N \subseteq X$ are proved.

$$LTR(M) \supseteq LTR(N) \Leftrightarrow (\forall x \in N)(\exists y \in N)\, y \preccurlyeq x \quad (7)$$

$$RLT(M) \subseteq RTL(N) \Leftrightarrow (\forall x \in M)(\exists y \in N)\, x \preccurlyeq y \quad (8)$$

Since the operators LTR and RTL can be applied for fuzzy sets, an ordering of fuzzy sets $A, B \in \mathcal{F}(X)$ with respect to crisp ordering $\preccurlyeq$ is generalized as:

$$A \preccurlyeq_I B \Leftrightarrow (LTR(A) \supseteq LTR(B) \wedge RTL(A) \subseteq RTL(B)) \quad (9)$$

The inclusion $LTR(A) \supseteq LTR(B)$ means that the left flank of *A* is to the left of the left flank of *B* while $RTL(A) \subseteq RTL(B)$ means that the right flank of *A* is to the left of the right flank of *B*.

Considering fuzzy orderings above, the fuzzy ordering calculation can be determined by considering horizontal positions of comparing fuzzy sets. If the assertion (9) is fulfilled in both conditions, the fuzzy ordering value is *true* or 1. Otherwise, the operation returns *false* or 0. Figure 2 shows the comparison of fuzzy sets that yields value 1.

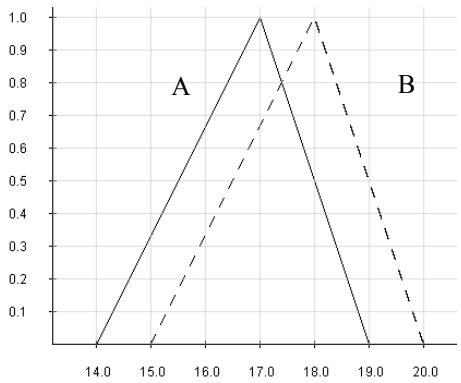


Figure 2. Comparison of fuzzy sets that satisfy (2)

From assertion (9) can be concluded that if only one condition is satisfied, it means that fuzzy sets cannot be compared - incomparable case. In this case, the fuzzy ordering operation will return *incomparable* or 0.5. Figure 3 shows the incomparable fuzzy sets.

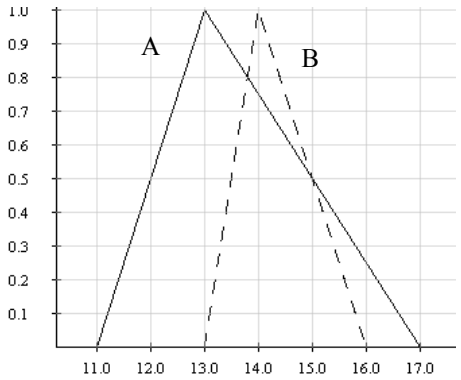


Figure 3. Incomparable fuzzy sets

Another incomparable case is the comparison of fuzzy sets having different heights. However, Skrbic and Rackovic proposed an idea to eliminate this problem in [5]. Fuzzy set A' is introduced as:

$$\mu_{A'} = \begin{cases} 1, \mu_A(x) = h(A) \\ \mu_A(x), otherwise \end{cases} \quad (10)$$

In this way, fuzzy relational operator $\leq_F$ is introduced by:

$$A \leq_F B \Leftrightarrow A' \leq'_F B' \quad (11)$$

Definition 4 Let A and B be two fuzzy sets over universe X. Order $\leq'_F$ over the set of all fuzzy sets over universe X, $\mathcal{F}(X)$ is defined by:

$$A \leq'_F B \Leftrightarrow (LTR(B) \subseteq LTR(A) \wedge RTL(A) \subseteq RTL(B)) \quad (12)$$

In the same way as with operators $<$ and $>$ on crisp domain, other relational operators, like $<_F$ and $>_F$ can be derived using the $\leq_F$ order.

C. Algorithm

As mentioned before, we consider five types of fuzzy set. Each type has different attributes that depict its properties. For example, a triangle fuzzy number contains three attributes (*LeftOffset*, *Maximum* and *RightOffset*). The *LeftOffset* refers to the beginning location of the support (*supp* in Definition 2) of fuzzy set (*LeftOffset*, 0). The *Maximum* refers to a location of its kernel (*Maximum*, 1) and the *RightOffset* refers to the end location of the support of fuzzy set (*RightOffset*, 0). Table I shows attributes for each type of characteristic function.

Attributes of fuzzy sets are used to calculate fuzzy relational operator values. Comparing two fuzzy sets, A and B, focuses on beginning, maximum and ending locations of A and B. For example, in Figure 1, two triangle fuzzy sets, A and B, are compared by operator $<$, the algorithm starts from comparing the *Maximum* attributes. If $Maximum_A$ is greater than $Maximum_B$, the result is 0 and the process ends. If not, the *LeftOffset* attributes will be compared. If $LeftOffset_A$ is not greater than $LeftOffset_B$, the process is still going onto compare *RightOffset*. If $RightOffset_A$ is greater than $RightOffset_B$, the result value is 0.5 (incomparable). If not, the result value is 1 (true). If $LeftOffset_A$ is greater than $LeftOffset_B$, $RightOffset_A$ and $RightOffset_B$ are compared. If $RightOffset_A$ is greater than $RightOffset_B$ then the result value is 0 (false), otherwise, the result value is 0.5 (incomparable). The algorithm for comparing two triangle fuzzy sets is shown in Listing 1.

TABLE I.
ATTRIBUTES OF EACH CHARACTERISTIC FUNCTION

Characteristic function	Attributes (A, μ_A)	Abbreviation
Triangle fuzzy number	LeftOffset $(A, 0)$	T-LO
	Maximum $(A, 1)$	T-MX
	RightOffset $(A, 0)$	T-RO
Trapezoidal fuzzy number	LeftOffset $(A, 0)$	TR-LO
	LeftMaximum $(A, 1)$	TR-LMX
	RightMaximum $(A, 1)$	TR-RMX
	RightOffset $(A, 0)$	TR-RO
Right shoulder	ZeroPoint $(A, 0)$	S-ZP
	Maximum $(\infty, 1)$	S-MX
Left shoulder	Maximum $(0, 1)$	S-MX
	ZeroPoint $(A, 0)$	S-ZP
Interval	LeftMaximum $(A, 1)$	I-LMX
	RightMaximum $(A, 1)$	I-RMX
Crip value	X (A)	C-X
	Y (μ_A)	C-Y

D. Crisp value

Unlike other fuzzy sets, the crisp value is a paired-value (A, μ_A). A comparison between crisp value and other fuzzy sets needs a special method.

For a relational operation between crisp value and another fuzzy set, we compare the value of attribute X of crisp value and boundary values of the compared fuzzy set. If a value X is less than the lower bound of the compared fuzzy set, the fuzzy ordering value is 1. If a value X is inside the boundary, the result value is 0.5. Otherwise, the result value is 0.

Comparing between crisp values is done in the same manner. For ordering between crisp values, A and B, following applies, if value X_A is not greater than value X_B, the result is 1. Otherwise the result is 0.

Listing 1. Algorithm for calculating fuzzy ordering between a triangle fuzzy number and another triangle fuzzy number.

```
Algorithm IsLessThan (FuzzyTriangle A, FuzzyTriangle B)
01. Compare Maximum_A and Maximum_B
02. If Maximum_A greater than Maximum_B
03.     Result is 0
04. Else
05.     Compare LeftOffset_A and LeftOffset_B
06.     If LeftOffset_A not greater than LeftOffset_B
07.         Compare RightOffset_A and RightOffset_B
08.         If RightOffset_A greater than RightOffset_B
09.             Result is 0.5
10.         Else
11.             Result is 1
12.         End if
13.     Else
18.         Compare RightOffset_A and RightOffset_B
19.         If RightOffset_A greater than RightOffset_B
20.             Result is 0
21.         Else
22.             Result is 0.5
23.         End if
23.     End if
24. End if
```

III. IMPLEMENTATION

To support our ideas, we developed the application that has two functions: manual fuzzy ordering testing and random fuzzy ordering testing. The manual testing function is used for a single test. In this case, the user can specify types of fuzzy sets and their attributes. When the process is done, the application shows an image of specified fuzzy sets and their fuzzy ordering value. Figure 4 illustrates the user interface for the manual testing function. The random testing function randomly generates comparison cases. In this function, the user can indicate types of fuzzy sets, number of generated cases, and boundary values. Figure 5 shows the user interface of the random testing function.

This application was developed on Java platform with the use of PostgreSQL to store fuzzy set attributes and cases of the random testing function.

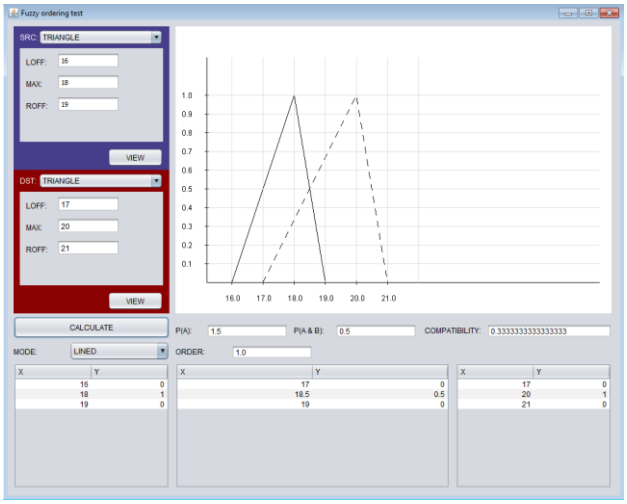


Figure 4. User interface of manual testing function

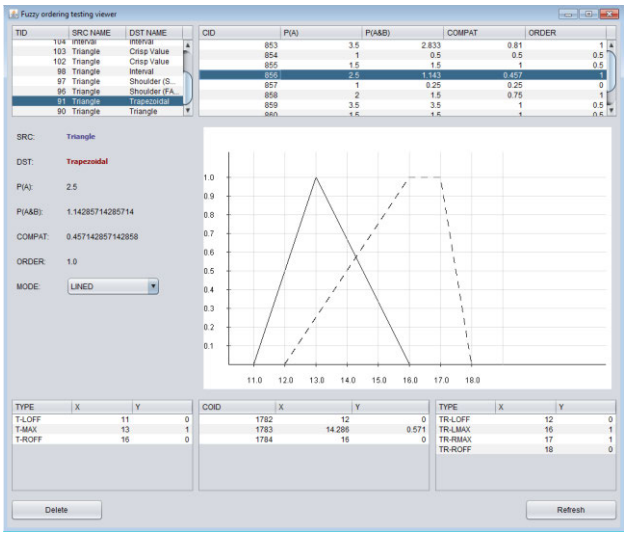
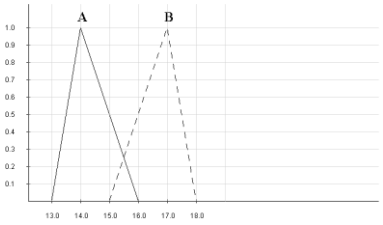
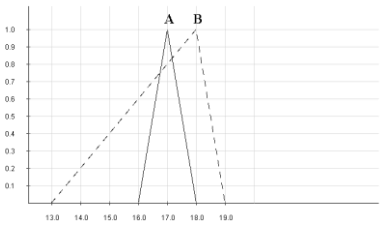
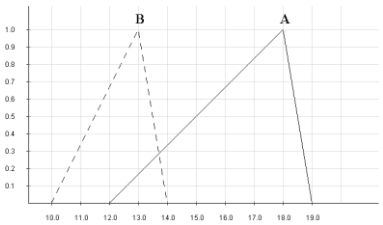
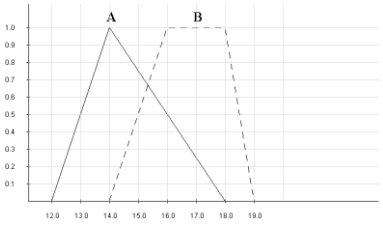


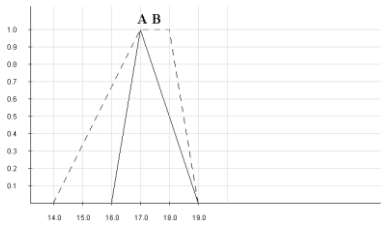
Figure 5. User interface of automated random testing function

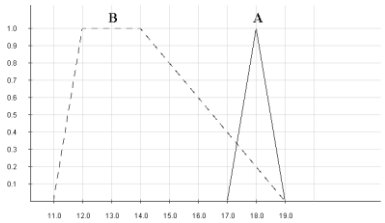
IV. TESTING RESULTS

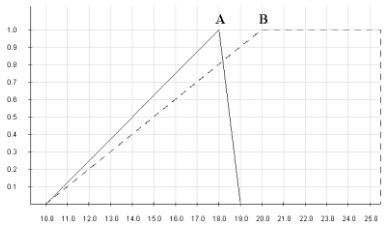
To prove the reliability of our proposed algorithms, some fuzzy ordering cases are generated randomly using random fuzzy ordering testing function of our application. As mentioned above, this paper considers five types of fuzzy sets. To cover all types of comparisons, each characteristic function is compared with the other four types including itself. Since there are two types of fuzzy shoulder, there are 36 comparison pairs. For a better variety in the comparison, the number of generated cases is set to be 10. Consequently, each pair has 10 cases of fuzzy ordering testing. Totally, there are 360 fuzzy ordering cases in our experimental results. The generated fuzzy sets are forced to position inside a boundary that is specified by the user. If the boundary is too wide, the fuzzy sets can be positioned too far from each other and have no incomparable cases. To avoid this problem, the lower bound and the upper bound are set to be 10 and 20, respectively. For the sake of brevity, some selected comparison cases between fuzzy triangle and other types are represented in Table II.

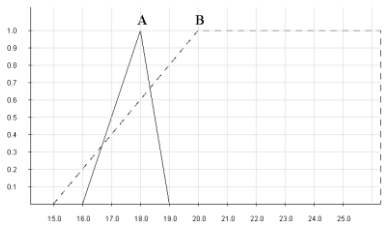
TABLE II.
A COMPARISONS BETWEEN FUZZY TRIANGLE AND OTHER TYPES

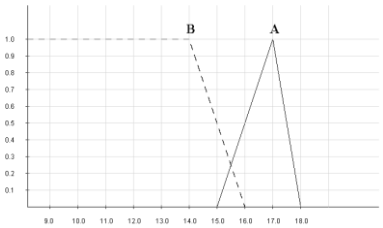
Case No.	Type of fuzzy set A		Type of fuzzy set B		Result
	Attributes	Value	Attributes	Value	
1	Fuzzy triangle		Fuzzy triangle		1
	$T\text{-}LO_A$	13	$T\text{-}RO_B$	15	
	$T\text{-}MX_A$	14	$T\text{-}MX_B$	17	
	$T\text{-}RO_A$	16	$T\text{-}RO_B$	18	
					
2	Fuzzy triangle		Fuzzy triangle		0.5
	$T\text{-}LO_A$	16	$T\text{-}RO_B$	13	
	$T\text{-}MX_A$	17	$T\text{-}MX_B$	18	
	$T\text{-}RO_A$	18	$T\text{-}RO_B$	19	
					
3	Fuzzy triangle		Fuzzy triangle		0
	$T\text{-}LO_A$	12	$T\text{-}RO_B$	10	
	$T\text{-}MX_A$	18	$T\text{-}MX_B$	13	
	$T\text{-}RO_A$	19	$T\text{-}RO_B$	14	
					
4	Fuzzy triangle		Fuzzy trapezoidal		1
	$T\text{-}LO_A$	12	$TR\text{-}LO_B$	14	
	$T\text{-}MX_A$	14	$TR\text{-}LMX_B$	16	
	$T\text{-}RO_A$	18	$TR\text{-}RMX_B$	18	
			$TR\text{-}RO_B$	19	
					

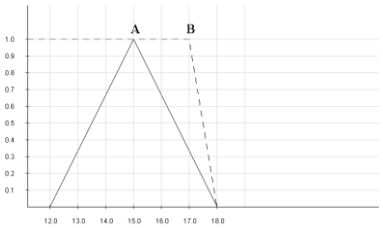
5	Fuzzy triangle		Fuzzy trapezoidal		0.5
	$T\text{-}LO_A$	16	$TR\text{-}LO_B$	14	
	$T\text{-}MX_A$	17	$TR\text{-}LMX_B$	17	
	$T\text{-}RO_A$	19	$TR\text{-}RMX_B$	18	
			$TR\text{-}RO_B$	19	
					

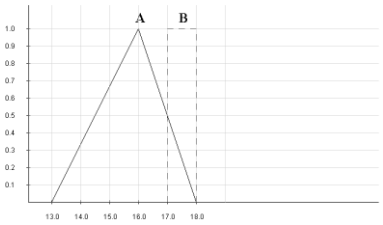
6	Fuzzy triangle		Fuzzy trapezoidal		0
	$T\text{-}LO_A$	17	$TR\text{-}LO_B$	11	
	$T\text{-}MX_A$	18	$TR\text{-}LMX_B$	12	
	$T\text{-}RO_A$	19	$TR\text{-}RMX_B$	14	
			$TR\text{-}RO_B$	19	
					

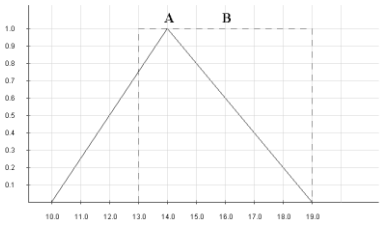
7	Fuzzy triangle		Right shoulder		1
	$T\text{-}LO_A$	10	$S\text{-}ZP_B$	10	
	$T\text{-}MX_A$	18	$S\text{-}MX_B$	20	
	$T\text{-}RO_A$	19			
					

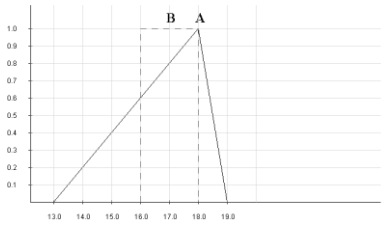
8	Fuzzy triangle		Right shoulder		0.5
	$T\text{-}LO_A$	16	$S\text{-}ZP_B$	15	
	$T\text{-}MX_A$	18	$S\text{-}MX_B$	20	
	$T\text{-}RO_A$	19			
					

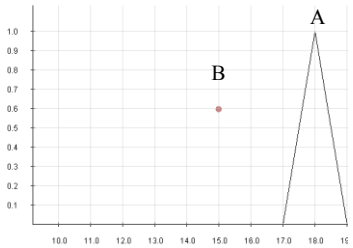
9	Fuzzy triangle		Left shoulder		0
	$T\text{-}LO_A$	15	$S\text{-}MX_B$	14	
	$T\text{-}MX_A$	17	$S\text{-}ZP_B$	16	
	$T\text{-}RO_A$	18			
					

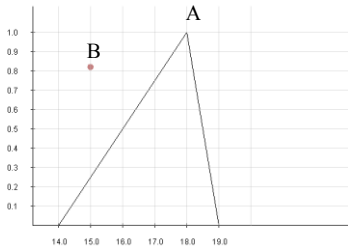
10	Fuzzy triangle		Left shoulder		0.5
	$T\text{-}LO_A$	12	$S\text{-}MX_B$	17	
	$T\text{-}MX_A$	15	$S\text{-}ZP_B$	18	
	$T\text{-}RO_A$	18			
					

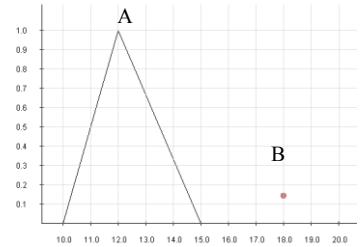
11	Fuzzy triangle		Interval		1
	$T\text{-}LO_A$	13	$I\text{-}LMX_B$	17	
	$T\text{-}MX_A$	16	$I\text{-}RMX_B$	18	
	$T\text{-}RO_A$	18			
					

12	Fuzzy triangle		Interval		0.5
	$T\text{-}LO_A$	10	$I\text{-}LMX_B$	13	
	$T\text{-}MX_A$	14	$I\text{-}RMX_B$	19	
	$T\text{-}RO_A$	19			
					

13	Fuzzy triangle		Interval		0
	$T\text{-}LO_A$	13	$I\text{-}LMX_B$	16	
	$T\text{-}MX_A$	18	$I\text{-}RMX_B$	18	
	$T\text{-}RO_A$	19			
					

14	Fuzzy triangle		Crisp value		0
	$T\text{-}LO_A$	17	C-X	15	
	$T\text{-}MX_A$	18	C-Y	0.595	
	$T\text{-}RO_A$	19			
					

15	Fuzzy triangle		Crisp value		0.5
	$T\text{-}LO_A$	14	C-X	15	
	$T\text{-}MX_A$	18	C-Y	0.819	
	$T\text{-}RO_A$	19			
					

16	Fuzzy triangle		Crisp value		1
	$T\text{-}LO_A$	10	C-X	18	
	$T\text{-}MX_A$	12	C-Y	0.143	
	$T\text{-}RO_A$	15			
					

V. CONCLUSION

This paper proposes the algorithm for binary fuzzy relational operators, which can be used to compare two fuzzy sets. Algorithms used to calculate fuzzy relational operator values are introduced. We developed an application that provides GUI and fuzzy relational operator calculations to prove the reliability of our algorithms. The testing results are generated randomly by this application. The results show that various comparisons are proved to be calculated correctly by our implementation. The proposed algorithms for fuzzy ordering will be used in FXI to enable comparison of two fuzzy sets.

Future research in this direction will tackle problems related to the implementation of aggregate functions, like MIN MAX, and SUM, using the proposed algorithms in FXI.

REFERENCES

[1] A. Champi, E. Damiani, S. Guinea, S. Marrara, G. Pasi, and P. Spoletini, "A Fuzzy Extension for the XPath Query Language" in *Flexible Query Answering Systems, Lecture Notes in Computer Science*, vol. 4027, 2006, pp. 210—221.

[2] E.J.T. Fredrick, and G. Radhamani, "Fuzzy Logic Based XQuery operations for Native XML Database Systems," in *International Journal of Database Theory and Application*, vol. 2, pp. 14—20.

[3] S. Skrbic, M. Rackovic, and M. Takaci, "Prioritized Fuzzy Logic Based Information Processing in Relational Databases," in *Knowledge-Based Systems*, vol. 38, 2013, pp. 62—73.

[4] P.S. Ueng, and S. Skrbic, "Implementing XQuery Fuzzy Extensions Using a Native XML Database," in *Proceeding of 13th IEEE International Symposium on Computational Intelligence and Informatics*, 2012, pp.305—309.

[5] S. Skrbic, and M. Rackovic, *Fuzzy databases*, Faculty of Sciences, University of Novi Sad, Novi Sad, 2013.

[6] U. Bodenhofer, "Orderings of Fuzzy Sets Based on Fuzzy Orderings Part I: The Basic Approach," in Mathware & Soft Computing, 2008, pp.201—218.