UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
DEPARTMENT OF MATHEMATICS AND INFORMATICS

DEVELOPMENT OF XQUERY INTERPRETER EXTENSIONS BASED ON FUZZY LOGIC WITH PRIORITIES

doctoral dissertation

RAZVOJ PROŠIRENJA XQUERY INTERPRETERA BAZIRAN NA FAZI LOGICI SA PRIORITETIMA

doktorska disertacija

Supervisor:
Dr. Srdjan Skrbic

Candidate:
Pannipa Sae-Ueng

Novi Sad, 2021

Contents

List of Figures

List of Tables

Listings

Abstract

In many real-world applications, information is often imprecise and uncertain. XML (eXtensible Markup Language) is one of the standards for data exchange over the internet, and with the popularity of web-based applications, huge amounts of data are available on the web. The XQuery is the language for querying XML data. However, XML and XQuery suffer from incapability of representing and manipulating imprecise and uncertain data. Consequently, this work explores options to represent fuzzy data in XML documents and extends XQuery language to provide a more flexible XQuery language by using the fuzzy set theory.

In this thesis, an extension of the XQuery query, called Fuzzy XQuery is described. It allows users to define priority, threshold and fuzzy expressions in their queries, and predefine linguistic terms to use them in querying. An algorithm for calculating the global constraint satisfaction degree using the Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) is introduced. Furthermore, Fuzzy XQuery Interpreter (FXI) is implemented allowing execution of fuzzy XQuery queries based on open-source technologies and native XML open-source database management system. Additionally, innovative methods for computing fuzzy set compatibility and introducing order over fuzzy sets have been implemented, which give serious improvements in computational performance compared to previous implementations.

Acknowledgements

This thesis would not have been possible without the assistance and support of numerous people. Many individuals have contributed their time and effort toward the completion of the study. Words cannot express my gratitude to those who have so kindly helped me. I would like to take this opportunity to acknowledge them and express my gratitude.

First and foremost, I would like to express my sincere gratitude to my academic advisor, Professor Dr. Srđan Škrbić, who shared with me his vision, gave me valuable suggestions, and who guided me through every step of the research. I am indebted to his excellent assistance, invaluable experience, and support throughout my studies. I truly appreciate his patience and tireless supervision.

I wish to pay my special regards to all the members of Thesis Presentation Committee and Thesis Defense Committee - Professor Dr. Miloš Racković, Assistant Professor Dr. Wiphada Wettayaprasit and Professor Dr. Aleksandar Takači, for their expert review, discussion and excellent comments. They provided different points of view, all of which were very helpful.

My deep appreciation goes to the Faculty of Science and Prince of Songkla University, Hat Yai Campus for financial support in my PhD study, and for offering me the invaluable opportunity to pursue a doctoral program.

I wish to show my gratitude to Assistant Professor Apirada Thadadach and Assistant Professor Dr. Supaporn Kansomkeat for the invaluable assistance that they all provided during my study. Moreover, my deep appreciation is extended to Sukgamon Sukpisit for his help with the implementation of the important parts of this work. My thanks are also due to Thomas

Daniel Houghton for his diligent proofreading of this Ph.D. thesis and Mintra Houghton for her help.

I would like to thank my sisters for their encouragement, love and help throughout the process. They enthusiastically provided me with continuous emotional support which has been invaluable at every step of my life. I thank my friends for their support throughout my graduate study.

Finally, and most importantly, I extend my heartfelt gratitude to my parents for their unconditional love, and for teaching me the importance of education and the pride that comes with accomplishment. I thank them for their patience, support and encouragement throughout this study.

Pannipa Sae-Ueng

Chapter 1

Introduction

1.1 The problem and the motivation

In many real-world applications, information is often imprecise and uncertain. For this reason, topics related to handle imprecise and uncertain information have been considered in the past.

With the popularity of web-based applications, the requirement has been put on the exchange and share of data over the web. The XML (eXtensible Markup Language) has become the de facto standard for data exchange over the internet. The query language to retrieve data stored in the form of XML is XQuery. The XQuery might be used to query XML databases in much the same way as we would use SQL (Structured Query Language) for relational databases. Unfortunately, XML and XQuery often suffer from incapability of representing and manipulating imprecise and uncertain data. Fuzzy set theory has been introduced as a successful technique for modelling fuzzy information in many application areas, especially in databases and XML documents. Consequently, we used the fuzzy set in this research to represent fuzzy data in XML documents and extended XQuery language as to provide a more flexible XQuery language.

In this work, we were inspired by the research of Skrbic et al., (Škrbić, Racković, 2013), (Škrbić, Racković, & Takači, 2011, 2013). They did many years of research related to fuzzy logic and fuzzy set theory extensions to relational databases. They modelled and implemented a set of tools that allow usage of fuzzy logic enriched with priorities in relational database applications. The relational data model was extended with the elements of fuzzy set theory. Moreover, they also defined a fuzzy extension of query language, called PFSQL (Priority Fuzzy SQL). Consequently, this approach was based on a similar idea, but we shifted towards extending a native XML database.

1.2 The results

The main contributions of this thesis are:

i) Fuzzy XQuery has been extended to allow users to define priority, threshold and fuzzy expressions in their queries.

ii) Extensions that include options to predefine linguistic terms in order to use them in queries.

iii) An implementation of the algorithm used to calculate the global constraint satisfaction degree by using the Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP), fuzzy compatibility, and fuzzy ordering functionalities in native XML database environments.

iv) The implementation of Fuzzy XQuery Interpreter (FXI) capable of executing the fuzzy XQuery queries based on open-source technologies and an open-source native XML database.

1.3 The structure of the thesis

The remainder of this thesis is organized as follows:

Chapter 2: Related Works reviews the main existing approaches in the area of fuzzy data representation in XML documents and XML query language flexibility.

Chapter 3: The Theoretical Background and Development Environment gives the basic concepts to be used in this thesis. It introduces the fuzzy set theory. Afterwards, the focus is moved on GPFCSP concept, fuzzy compatibility and fuzzy ordering. Next, the concepts of XML, XML databases are explained, including a presentation of eXist-db –native XML database used in this thesis. Finally, technologies used for implementation - ANTLR (ANother Tool for Language Recognition), Spring Boot and RESTful web services are presented.

Chapter 4: System implementation describes the details of implementation in five sections: (i) Fuzzy XQuery EBNF grammar provides the fuzzy XQuery syntax in EBNF (Extended Backus Normal Form) notation, (ii) the representation of fuzzy data, (iii) the use case diagram of FXI system, (iv) the algorithm and fuzzy XQuery processing, and (v) development of the system and GUI (Graphical User Interface).

Chapter 5: Software Testing examines the correctness and performance testing of the FXI system.

Chapter 6: Conclusions provides a summary of the work presented in this thesis.

Appendix A: Complete EBNF grammar of fuzzy XQuery

Appendix B: Test cases information

Chapter 2

Related Works

This chapter gives the literature review of attempts to introduce fuzzy mechanisms into XML technologies. In the last decade, we observed that there were two main research directions on storage and querying of imprecise and uncertain XML data. The first one was to study ways to represent fuzzy data in XML documents. The second one was devoted to achieving flexibility in XML querying languages.

2.1 Fuzzy data in XML documents

There have been many different approaches to represent uncertain information in XML documents.

Üstünkaya et al. (Üstünkaya, Yazici, & George, 2007) focused on the fuzzy-XML documents containing fuzzy-valued attributes which may have various semantics as "OR", "XOR" and "AND" for relating the fuzzy values.

Jin and Veerappan (2010) presented the implementation that supported the linguistic label and approximate values. The linguistic label defines the linguistic variable from a trapezoidal and interval distribution, while approximate values have been defined by a triangular distribution only.

The work of Oliboni and Pozzani (2008) proposed an XML schema definition for representing different aspects of fuzzy information. They classified fuzzy data types into four types in the fuzzy XML documents, which are *classicType*, *FuzzyOrdType*, *FuzzyNonOrdSimType* and *FuzzyNonOrdType*. Moreover, they defined three kinds of degrees, which are *FuzzyAttrDegree*, *FuzzyInstDegree* and *FuzzyNonAssDegree*.

Panić et al. (Panić, Racković, & Škrbić, 2014) proposed the indefiniteness in XML documents, which are the indefiniteness in XML values and the indefiniteness in XML structure. Indefiniteness in XML values is achieved by introducing two types of elements: *fuzzy* and *function*. Element *fuzzy* indicates a fuzzy set defined by a membership function, while element *function* is used to define an arbitrary membership function under the element *fuzzy*. Indefiniteness in structure is achieved by introducing the element *possibility*, which represents the probability of an element occurring in an XML document.

Yan et al. (Yan, Ma, & Zhang, 2014) presented the fuzzy construct *Val* and *Dist*. The fuzzy construct *Val* was used to specify the possibility of an element in a fuzzy XML document that is paired with a *possibility* attribute like <Val Poss=0.8> and </Val>, while the fuzzy construct *Dist* was used to specify the type of distribution: disjunctive and conjunctive. The former represents a set of possible values from which the only one is true, while the latter represents a set of possible values that all "true" with various degrees.

2.2 Flexibility in XML query languages

Several approaches have been proposed for adding flexibility to XML query languages. We concluded that they could be categorized into the following four categories: XQuery-based, XPath-based, algebra-based and twig-based.

2.2.1 XQuery-based

Lo et al. (Lo, Kianmehr, Kaya, Ozyer, & Alhajj, 2007) built a tool that introduces flexibility to XQuery queries. They implemented the VIREX tool that a user could use to specify membership functions of fuzzy terms in a database table. However, it supported only triangular distribution. Queries containing fuzzy data are translated into corresponding ordinary XQuery queries.

To support fuzziness in XQuery, Goncalves and Tineo (2007, 2010) defined a new *xs:truth* data type to represent truth degrees and a new *xml:truth* attribute to handle satisfaction degree in the node of fuzzy XQuery expressions. The fuzzy terms were declared and used in query expressions. Additionally, they proposed an evaluation algorithm for XQuery queries based on the Derivation Principle (Goncalves & Tineo, 2005). Moreover, Labbad et al. (Labbad, Monascal, & Tineo, 2016) developed an implementation of Fuzzy XQuery that was proposed by Goncalves and Tineo's concepts.

Jin and Veerappan (2010) presented the fuzzy query language based on the XQuery standard that allowed fuzzy expressions with various conditions in a query. This approach allowed both crisp and fuzzy data in the XML document as in our approach. Thus, two cases are possible for comparison: 1) one value is crisp and the other one is fuzzy and 2) both values are fuzzy. The query is processed by comparing two ranges of values for overlapping of two fuzzy distributions with the comparison translation rules.

Fredrick and Radhamani (2010, 2011) introduced the algorithm for fuzzy XQuery processing for native XML databases and implemented a querying tool with a GUI using VB.Net. They allowed the users to use linguistic terms in an XQuery query. The linguistic terms refer to fuzzy membership functions based on fuzzy sets. If the values are in the fuzzy membership range, then the native XML database returns XML data.

A similar approach as presented in this work was proposed by Panić et al. (2014). They expanded XQuery syntax with fuzzy values and included priorities and thresholds in fuzzy XQuery extensions using the GPFCSP concept in the same way as we did. However, there are many differences between Panić's work and this work. Firstly, Panić's implementation used .NET framework, MATLAB and the Microsoft SQL Server database in the Windows-based application, whereas our approach uses open-source technologies - Java programming language to implement the new interpreter that is independent of MATLAB with eXist-db – a native XML database and web-based application for user access. The core methods of implementation of algorithms for GPFCSP based satisfaction degree calculations, fuzzy compatibility calculations and fuzzy ordering calculations are completely different. Moreover, we used completely different ways to represent fuzzy data in an XML document.

2.2.2 XPath-based

Fuzzy versions of XPath have been previously studied in some works. For example, Amer-Yahia et al. (Amer-Yahia, Lakshmanan, & Pandit, 2004) described the FleXPath that integrates XPath querying with full-text search in a flexible way. It uses 'template' query expressions and seeks answers that are approximation matches to this template. They introduced fuzziness by query relaxations and defined four operations: axis generalization, left deletion, subtree promotion and *contains* promotion on the structure of queries. The efficient three algorithms were developed for answering top-k queries: Dynamic Penalty Order (DPO), Static Selectivity Order (SSO) and Hybrid.

Campi et al. (2009), (Campi, Guinea, & Spoletini, 2014) proposed FuzzyXPath that has a *deep-similar* function. This function provides a degree of similarity between two XML trees for assessing whether they are both similar.

Almendros et al. (Almendros-Jiménez, Tedesqui, & Moreno, 2015) presented an extension of the XPath query language that provided ranked answers. They proposed a fuzzy variation of *and*, *or* and *avg* operators for XPath conditions as well as two structural constraints called *down* and *deep*. They also implemented an application by using the Fuzzy Logic

Programming Environment for Research (FLOPER) tool which was based on Multi Adjoint Logic Programming (MALP).

2.2.3 Algebra-based

Ma et al. (Ma, Li, & Yan, 2010) proposed a formal fuzzy XML query algebra for expressing fuzzy XML queries by transforming the XQuery expressions into algebraic expressions. They also introduced the fuzzy XML algebraic operations: *set operations*, *fuzzy selection*, *fuzzy projection*, *fuzzy join*, *fuzzy grouping*, *fuzzy ordering*, and *bind and tree*, to apply the algebra to XML query processing.

2.2.4 Twig-based

Yan et al. (2014) proposed an XML query formed as a twig pattern with predicates additionally imposed on the contents or attribute values of the tree nodes. They presented three types of fuzzy XML twig queries with AND, OR and NOT predicates, and used the fuzzy extended Dewey encoding to encode the XML document tree. Li and Ma (2018) introduced the structure query language into the keyword query in fuzzy XML data to get more comprehensive query results. First, they proposed the concepts of *object tree*, *the minimum object tree*, *the nearest object tree* and proposed semantics of matching object tree for a keyword query. Then, they gave their query method AO-Twig to combine the structure query language with keyword query to obtain the Top-K query results with the highest scores.

In addition, there was an effort that did not classify to any categories such as Üstünkaya et al. (2007). They defined the fuzzy attribute in XML documents that were kept in Tamino (a native XML database) and allowed users to perform fuzzy queries by specifying fuzzy attributes and threshold values. The similarity matrix was used to find results matching fuzzy-value attributes.

The three aspects of comparative details between corresponding approaches and our work in terms of how fuzziness was achieved in XQuery and how various details were implemented are presented in Table 2.1.

The first aspect is support for fuzziness in XQuery. Several approaches as in Panić et al. (2014), Labbad et al. (2016), Fredrick & Radhamani (2011), Jin & Veerappan (2010) and this work allowed users to define fuzzy terms in a query. A fuzzy term is an expression that a user can define as a linguistic label that is interpreted based on distributions of membership functions according to the user's preferences. Moreover, a user can define a threshold value.

Results that have a membership degree less than the defined threshold are removed from result set as in Panić et al. (2014), Labbad et al. (2016), Üstünkaya et al. (2007) and our work. In addition, Panić et al. (2014) and our approach support priority expressions to specify the importance of each fuzzy condition. To support fuzziness in XQuery, it is necessary to calculate a satisfaction degree, which is a real number between 0 and 1, for the result tuples satisfying fuzzy conditions, as presented in Panić et al. (2014), Labbad et al. (2016), and this work. One of the advantages of our approach is the fuzzy ordering function implementation used to compare values between two fuzzy sets (see chapter III for more details).

The second aspect is the strategy for fuzzy query processing. There are various ways and mechanisms to process evaluation of fuzzy XQuery results such as using: Generalized Prioritized Fuzzy Constraint Satisfaction Problem: GPFCSP (described in detail in chapter III), Derivation Principle, Fuzzy membership range, similarity matrix and Comparison Translation Rules.

The third aspect is the implementation. Most implementations have used an open source programming language (Java) and a native XML database such as eXist-db, Oracle and Tamino. Our work is the first implementation that offers a web application based on eXist-db.

Table 2.1 Comparing approaches (Sae-Ueng & Skrbic, 2020)

Papers	Fuzziness in XQuery					Theory	Implementation		
	Fuzzy term	Priority	Threshold	Satisfaction degree	Ordering		DB	Programming language	Application base
Current approach	✓	✓	✓	✓	✓	GPFCSP	eXist-db	JAVA	Web-based
Panić et al.	✓	✓	✓	✓	x	GPFCSP	MS SQL Server	.NET	Windows-based
Labbad et al.	✓	x	✓	✓	x	Derivation Principle	eXist-db	JAVA	Client-Server
Fredrick and Radhamani	✓	x	x	x	x	Fuzzy membership range	eXist-db	VB.NET	Unknown
Üstünkaya et al.	x	x	✓	x	x	Similarity matrix	Tamino	JAVA	Standalone application
Jin & Veerappan	✓	x	x	x	x	Comparison Translation Rules	Oracle Berkeley DB XML	JAVA	Unknown

Chapter 3

Theoretical Background and Development Environment

This chapter presents theory and methods used later in the thesis or related to the topic of the thesis. First, we present the theory of fuzzy sets in section 3.1. Next, we give details of the Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) in section 3.2. The Fuzzy Compatibility and Fuzzy ordering are described in section 3.3 and section 3.4, respectively. Section 3.5 provides the eXtensible Markup Language (XML) concept. The XML database and eXist-db database are presented in section 3.6 and section 3.7, respectively. Section 3.8 describes the ANTLR (ANother Tool for Language Recognition) tool. Spring Boot is briefly presented in section 3.9, while options for building the RESTful web services and RESTful API in eXist-db are explained in section 3.10 and 3.11.

3.1 Fuzzy sets

The fuzzy theory was proposed by Zadeh (1965). A fuzzy set can be defined as follows. A fuzzy set *A* over universe X is determined by its characteristic (membership) function

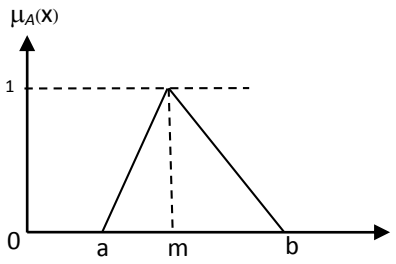
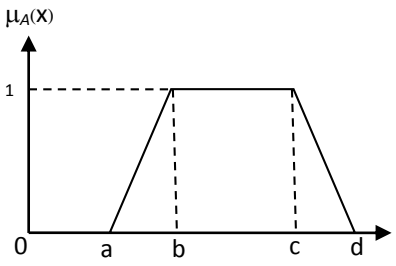
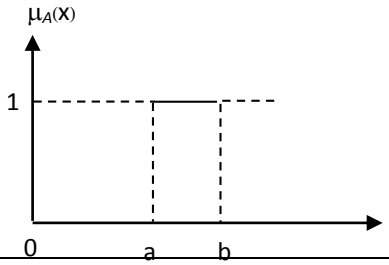
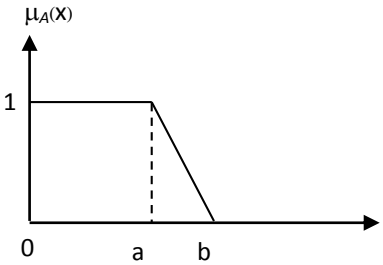
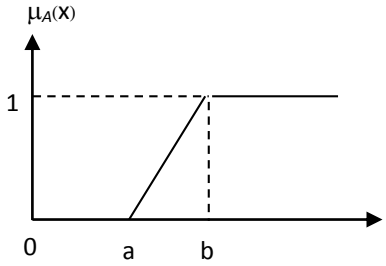
$$\mu_A : x \rightarrow [0,1]$$

where, for every $x \in X$, μ_A is a degree of membership of element x to the fuzzy set *A*. The characteristic functions of fuzzy numbers used in this work are given in table 3.1.

When the fuzzy numbers represent linguistic concepts, such as low, medium and expensive as interpreted in a particular context, the resulting constructs are usually called *linguistic variables*.

A linguistic variable is a variable whose values are not numbers but words or sentences in a natural or artificial language. The main purpose of using linguistic values (words or sentences) instead of numbers is that linguistic characterizations are less specific than numerical ones but much closer to the way that humans express and use their knowledge. For example, if we say "this book is expensive" is less specific than "the book is 300 €". In that case, "expensive" can be seen as a linguistic value of the variable "price" which is less precise and informative than the numerical value "300".

Table 3.1 Characteristic functions (Alonso, n.d.)

Type	Membership Function	Two-dimensional graph
Triangular fuzzy number	$\mu_A(x) = \begin{cases} \frac{(x-a)}{(m-a)}, x \in (a,m], \\ \frac{(b-x)}{(b-m)}, x \in (m,b), \\ 0, otherwise; \end{cases}$	
Trapezoidal fuzzy number	$\mu_A(x) = \begin{cases} \frac{(x-a)}{(b-a)}, x \in (a,b), \\ 1, x \in [b,c] \\ \frac{(d-x)}{(d-c)}, x \in (c,d), \\ 0, otherwise; \end{cases}$	
Interval	$\mu_A(x) = \begin{cases} 0, x < a, x > b, \\ 1, x \in [a,b], \end{cases}$	
Fuzzy shoulder-Left shoulder	$\mu_A(x) = \begin{cases} 1, x <= a, \\ \frac{(b-x)}{(b-a)}, x \in (a,b), \\ 0, otherwise; \end{cases}$	
Fuzzy shoulder-Right shoulder	$\mu_A(x) = \begin{cases} 0, x <= a, \\ \frac{(x-a)}{(b-a)}, x \in (a,b), \\ 1, otherwise; \end{cases}$	

Each linguistic variable is expressed by linguistic terms interpreted as specific fuzzy numbers. An example of a linguistic variable "Price" is shown in figure 3.1. This variable

expresses the price of a book in a given context by three linguistic terms: low, medium and expensive. Each of the linguistic terms is assigned one of five fuzzy numbers by a semantic rule, as shown in the figure.

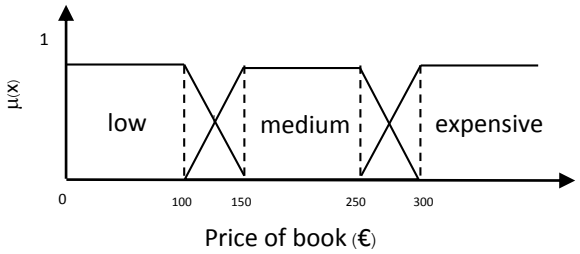


Figure 3.1 The linguistic variable "Price"

3.2 Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP)

The concept of Constraint Satisfaction Problem (CSP) is the problem defined as a set of objects whose states must satisfy a number of constraints or limitations. The CSP can be extended to the Fuzzy Constraint Satisfaction Problem (FCSP) by modelling constraints as fuzzy sets over a particular domain. The Priority Fuzzy Constraint Satisfaction Problem (PFCSP) (Dubois, Fargier, & Prade, 1996) is a type of FCSP that introduces the notion of priority. In this way, the value of a 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 (2005) generalized the PFCSP to the GPFCSP by adding the possibility to use disjunction and negation whereas the definition of the GPFCSP is an extension of the PFCSP definition. We recall the definition of PFCSP and GPFCSP here.

Definition 1: A Priority Fuzzy Constraint Satisfaction Problem (PFCSP) is a quadruple (X,D,C^f, ρ) where

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

2. $D = \{d_i | i=1,2,\ldots,n\}$ is a set of domains. Every domain d_i is a finite set of possible values for the corresponding variable x_i in X,

3. C^f is a finite nonempty set of elements called fuzzy constraints, that is

$$C^f = \{R_i^f | \mu_{R_i^f} : (\prod_{x_j \in var\,(R_i^f)} d_j) \rightarrow [0, 1], i=1, 2,\ldots,n\},$$

where $var(R_i^f)$ is a set of variables contained inside the constraint R_i^f

4. $\rho \in C^f \rightarrow [0,\infty)$

and a combined valuation v_x of all variables in X, and $\oplus_\rho : [0,1]^n \rightarrow [0,1]$, $g:[0,\infty]x[0,1] \rightarrow [0,1]$, and a global satisfaction degree $\alpha_\rho(v_x)$ which is calculated as

$$\alpha_\rho(v_x) = \oplus_\rho \left\{ g(\rho(R^f), \mu_{R_i^f}\left(v_{var(R^f)}\right) \middle| R^f \in C^f \right\}$$

This system is a PFCSP if the following axioms are satisfied:

1. If for the fuzzy constraint R^f_{max}, we have

$$\rho_{max} = \rho(R^f_{max}) = \max\{\rho(R^f) \mid R^f \in C^f\},$$

then

$$\mu_{R^f_{max}}(v_x) = 0 \Rightarrow \alpha_\rho(v_x) = 0$$

2. If $\forall R^f \in C^f$, $\rho(R^f) = \rho_0 \in [0,1]$, then

$$\alpha_\rho(v_x) = \oplus_\rho \left\{ \mu_{R_i^f}\left(v_{var(R^f)}\right) \middle| R^f \in C^f \right\},$$

Where $\oplus_\rho$ is a triangular norm.

3. For $R_i^f, R_j^f \in C^f$, assume $\rho(R_i^f) \geq \rho(R_j^f)$, $\delta > 0$ and there are two different valuations v_x and v'_x such that:

3.1 if $\forall R^f \neq R_i^f$ and $\forall R^f \neq R_j^f$, then $\mu_{R^f}(v_{var(R^f)}) = \mu_{R^f}(v'_{var(R^f)})$,

3.2 if $R^f = R_i^f$, then $\mu_{R^f}(v_{var(R^f)}) = \mu_{R^f}(v'_{var(R^f)}) + \delta$,

3.3 if $R^f = R_j^f$, then $\mu_{R^f}(v'_{var(R^f)}) = \mu_{R^f}(v_{var(R^f)}) + \delta$.

then if

$$g\left(\rho(R_i^f), \mu_{R_i^f}\left(v_{var\left(R_i^f\right)}\right)\right) \leq g\left(\rho(R_j^f), \mu_{R_j^f}\left(v_{var\left(R_j^f\right)}\right)\right)$$

then $\alpha_\rho(v_x) \geq \alpha_\rho(v'_x)$ holds.

4. For two different combined labels v_x and v'_x such that $\forall R^f \in C^f$, if

$$\mu_{R^f}(v_{var\left(R^f\right)}) \geq \mu_{R^f}(v'_{var(R^f)})$$

then $\alpha_\rho(v_x) \geq \alpha_\rho(v'_x)$ holds.

5. If there exists a combined valuation v_x and v'_x such that $\forall R^f \in C^f$, $\mu_{R^f}(v_{var(R^f)}) = 1$ then $\alpha_\rho(v_x)=1$.

Definition 2: Let X, D, C^f, ρ, g and v_x be defined as in definition 1. The generalized PFCSP is defined as a tuple (X, D, C^f, ρ, g, ∧, ∨, ¬) which satisfies the following.

An elementary formula in GPFCSP is a pair(x, $\rho(C_i)$), where $C_i \in C^f$, $x \in Dom(C_i)$ represents the satisfaction degree of a constraint C_i and $\rho_i= \rho(C_i)$, represents its priority.

A formula in GPFCSP is defined in the following way,

1. An elementary formula is a formula.

2. If f_1 and f_2 are formulas then also ∧(f_1, f_2), ∨(f_1, f_2) and ¬(f_1) are formulas.

For each valuation v_x, a satisfaction degree $\alpha_F(v_x)$ of a formula F is calculated depending on the interpretation of connectives.

A system is a GPFCSP if:

1. Let $F=\wedge_{i\in\{1,\ldots,n\}} f_i$ be a formula in the GPFCSP where f_i, $i\in\{1,\ldots,n\}$ are elementary formulas and let be a set of constraints that appear in the formula. If for the fuzzy constraint R^f_{max}we have

$$\rho_{max} = \rho(R^f_{max}) = \max\{\rho(R^f) \mid R^f \in C^f\},$$

Then for each formula F we have:

$$\mu_{R^f_{max}}(v_x)= 0 \Rightarrow \alpha_F(v_x) = 0$$

2. If $\forall R^f \in C^f$, $\rho(R^f) = \rho_0 \in [0,1]$, then for each formula F the following holds:

$$\alpha_F(v_x) = F_{£}(v_x),$$

Where $F_£$ is the interpretation of the logical formula F in fuzzy logic £(∧, ∨, ¬).

3. For $R^f_i, R^f_j \in C^f$, assume $\rho(R^f_i) \geq \rho(R^f_j)$, $\delta > 0$ and assume that there are two different valuations v_x and v'_x such that:

3.1 if $\forall R^f \neq R^f_i$and $\forall R^f \neq R^f_j$, then $\mu_{R^f}(v_x) = \mu_{R^f}(v'_x)$,

3.2 if $R^f = R^f_i$, then $\mu_{R^f}(v_x) = \mu_{R^f}(v'_x)+ \delta$,

3.3 if $R^f = R^f_j$, then $\mu_{R^f}(v'_x) = \mu_{R^f}(v_x)+ \delta$.

Then, for formulas:

$$F= \bigwedge_{k=1,\ldots,n}(x_k, \rho(R_k)),\ x_k \in Dom(R_k)$$

and

$$F= \bigvee_{k=1,\ldots,n}(x_k, \rho(R_k)),\ x_k \in Dom(R_k)$$

The following holds:

$$\alpha_F(v_x) \geq \alpha F(v'_x)$$

4. Assume that two different compound labels v_x and v'_x such that $\forall R^f \in C^f$ satisfy

$$\mu_{R^f}(v_x) \geq \mu_{R^f}(v'_x)$$

If formula F has no negation connective, then the following holds:

$$\alpha_F(v_x) \geq \alpha_F(v'_x)$$

5. Let there be a compound label such that $\forall R^f \in C^f$, $\mu_{R^f}(v_x) = 1$. If F is a formula $F = \bigwedge_{k=1,\ldots,n} f_i$ or $F = \bigvee_{k=1,\ldots,n} f_i$, where f_i, $i \in \{1,\ldots,n\}$ are elementary formulas then

$$\alpha_F(v_x) = 1$$

To obtain the global satisfaction degree, we need to aggregate the value of each constraint. We can use aggregation operators from fuzzy logic t-norms, t-conorms and fuzzy negation as shown in the GPFCSP definition below. The proof can be found in Takaci, Skrbic, & Perovic (2009). Here we present a definition that describes one practically usable GPFCSP system.

Theorem: the following system $(X, D, C^f, \rho, g, \wedge, \vee, \neg, \Diamond)$ where $\wedge = T_L$, $\vee = S_L$, $\neg = N_s$, and $\Diamond(x_i, c_i) = S_p(x_i, 1-\rho_i)$ is a GPFCSP. The global satisfaction degree of a valuation v_x for a formula F is obtained in the following way:

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

Where C^f is the set of constraints of formula F, $\rho_{max} = \max\{\rho(R_i^f), R^f \in C^f\}$, T_L is the Łukasiewicz triangular norm, S_L is the Łukasiewicz triangular conorm, N_s is the standard negation (1-x), and S_p is the triangular product conorm.

Global satisfaction degree calculation for a given valuation will be illustrated in an example.

Example Let us observe a set of students from the faculty of Science, Prince of Songkla University. We will rank them using two criteria – age and height (in centimetres). Let us construct the variables and their domains in the following way:

1. X_1 represents age, $D_1 = [0,100]$
2. X_2 represents height, $D_2 = [100,250]$

We also define the following constraints:

1. R_1^f = "young students"
2. R_2^f = "tall students"

Constraints are fuzzy subsets of the corresponding domains as in figure 3.2. The first constraint is modelled as a left-shoulder with the upper offset and lower offset of 20 and 25, respectively. The second constraint is a right-shoulder that ascends from 170 to 180.

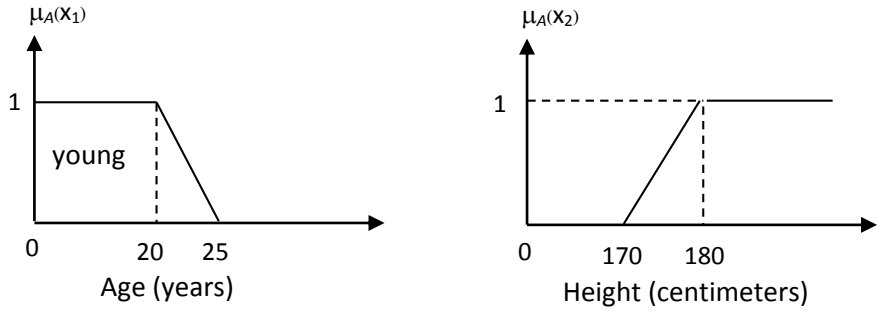


Figure 3.2 R_1^f and R_2^f membership functions

Suppose that we retrieve the students who have a young age (R_1^f) and about 180 cm height (R_2^f). In addition, the priority of each constraint is represented like this: $\rho(R_1^f) = 0.6$ and $\rho(R_2^f) = 0.3$. Valuation v_x given in table 3.2 defines three students.

Table 3.2 Valuation v_x

Name	Age	Height
Mary	20	180
Peter	18	160
John	25	175

First, we calculate constraint satisfaction degree ($\mu(R_i^f)$) for every constraint and every student. These degrees are obtained directly as values of corresponding membership functions in given points. Results are given in table 3.3.

Table 3.3 Constraint satisfaction degree

Name	$\mu(R_1^f)$	$\mu(R_2^f)$
Mary	1	1
Peter	1	0
John	0	0.5

Now we can calculate the global constraint satisfaction degree for every student in the following way:

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

If we use values for the first student, we obtain the following:

$$\alpha = T_L(S_P(1,1-0.6), S_P(1,1-0.3)) = T_L(S_P(1,0.4), S_P(1,0.7))$$

The definition of the Łukasiewicz triangular norm (T_L) and triangular product conorm (S_P) as following:

$$T_L(x,y) = \max(x+y-1,0)$$

$$S_P(x,y) = x+y-xy$$

Finally, we obtain satisfaction degree for the first student:

$$\alpha = T_L(1,1) = 1$$

Global constraint satisfaction degrees for all students are calculated in the same way and given in table 3.4. These results are a measure of how much does every one of the students satisfy our criteria.

Table 3.4 Global constraint satisfaction degree for all students

Name	α
Mary	1
Peter	0.7
John	0.91

In a way, GPFCSP systems can be made similar to FLWOR clause of XQuery. The 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. *Where* clause contains a sequence of constraints connected with logical operators in much the same way as in GPFCSP. The example of the FLWOR clause is shown in Listing 3.1.

Listing 3.1 An example of the FLWOR clause

```
for $x in document(student.xml)
where $x/height>180 AND $x/age = 20
return $x
```

3.3 Fuzzy Compatibility

In fuzzy XQuery statements, variables can assume fuzzy or non-fuzzy values. Normally, different types of values cannot be compared directly. Therefore, it is necessary to implement a fuzzy compatibility calculation to solve this problem. The equation of compatibility of fuzzy set A to the fuzzy set B is given below (Škrbić, Racković, 2013).

$$\text{Compatibility}_{A,B} = \frac{P(A \cap B)}{P(A)} \qquad \text{(E1)}$$

$P(A \cap B)$ 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 three steps (Sukpisit, Kansomkeat, Sae Ueng, Thadadech, & Škrbić, 2016):

1. Calculate the intersection area,
2. Calculate the size of obtained intersection area,
3. Calculate the compatibility value.

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

In step 2, the cyclic polygon calculation proposed by Pak (Pak, 2005) will be used to calculate the size of the intersection area. This method uses coordinates of a polygon for the area calculations. The area is calculated by the following equation:

$$\text{Area} = \left| \frac{(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + \cdots + (x_ny_1 - x_1y_n)}{2} \right| \qquad \text{(E2)}$$

In step 3, a compatibility value can be obtained by equation (E1) using the size of the intersection area provided by step 2 and the size of the area of the source fuzzy set that can be calculated by the equation (E2). A more detailed explanation of the fuzzy compatibility calculation is given in chapter 4.

3.4 Fuzzy Ordering

When the relational operators (<,<=,>,>=) are included in the query, it is necessary to provide means to calculate comparison values between fuzzy sets. These fuzzy relational operators are typically used in two fuzzy set comparison cases but can also be used with some aggregate functions like MIN, MAX, and SUM. A well-known ordering for real intervals is:

$$[a, b] \preccurlyeq_I [c, d] \Leftrightarrow a \leq c \,\&\, b \leq d$$

The inequality $a \leq c$ means that there are no elements of the set [c,d] that are below the interval [a,b]. While the inequality $b \leq d$ means that there are no elements of [a,b] that above [c,d].

Bodenhofer (2008) has introduced an ordering as given below:

Definition 3 Let A be a fuzzy set of the domain X, a fuzzy superset of A, denoted by LTR(A) (standing for Left-To-Right closure) is the smallest fuzzy superset of A that has a non-decreasing membership function (see figures 3.3) as defined by:

$$\text{LTR}(A)(x) = \sup\{A(y) \mid y \in X \,\&\, y \leq x\}$$

Definition 4 A fuzzy superset of A, denoted by RTL(A) (standing for Right-To-Left closure) is the smallest fuzzy superset of A that has a non-increasing membership function (see figures 3.3) as defined by:

$$\text{RTL}(A)(x) = \sup\{A(y) \mid y \in X \,\&\, x \leq y\}$$

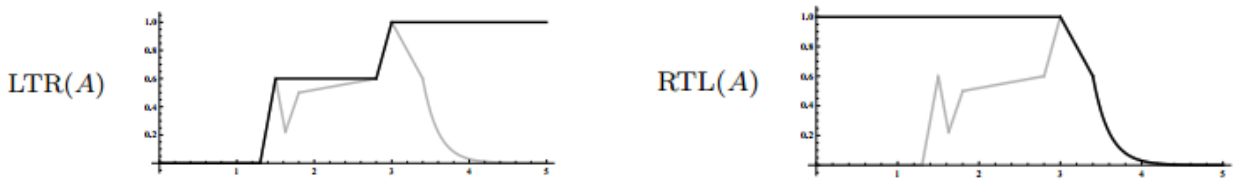


Figure 3.3 LTR(A) and RTL(A) (Bodenhofer, 2008)

Definition 5 The F(X) denotes the set of all fuzzy sets of X, a fuzzy ordering on fuzzy set A and B, if A, B ∈F(X), is defined as follows:

$$A \preccurlyeq_I B \Leftrightarrow (LTR(A) \supseteq LTR(B) \,\&\, RTL(A) \subseteq RTL(B)) \qquad (E3)$$

Considering fuzzy orderings above, the fuzzy ordering calculation can be determined by considering horizontal positions of compared fuzzy sets (Kansomkeat, Sukpisit, Thadadech, Sae Ueng, & Skrbic, 2015). 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) we can conclude 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.

3.5 eXtensible Markup Language (XML)

XML is a data formatting recommendation proposed by the W3C as a simplified form of the Standard Generalized Markup Language (SGML) - one of the standards for data description and exchange between various systems and databases over the Internet. As a markup language, XML supports user-defined tags, encourages the separation of document content from its presentation, and can automate web information processing. Figure 3.4 shows the relationship of XML documents, DTD, XSD, XQuery, XPath and XSLT that will be described in the next sections.

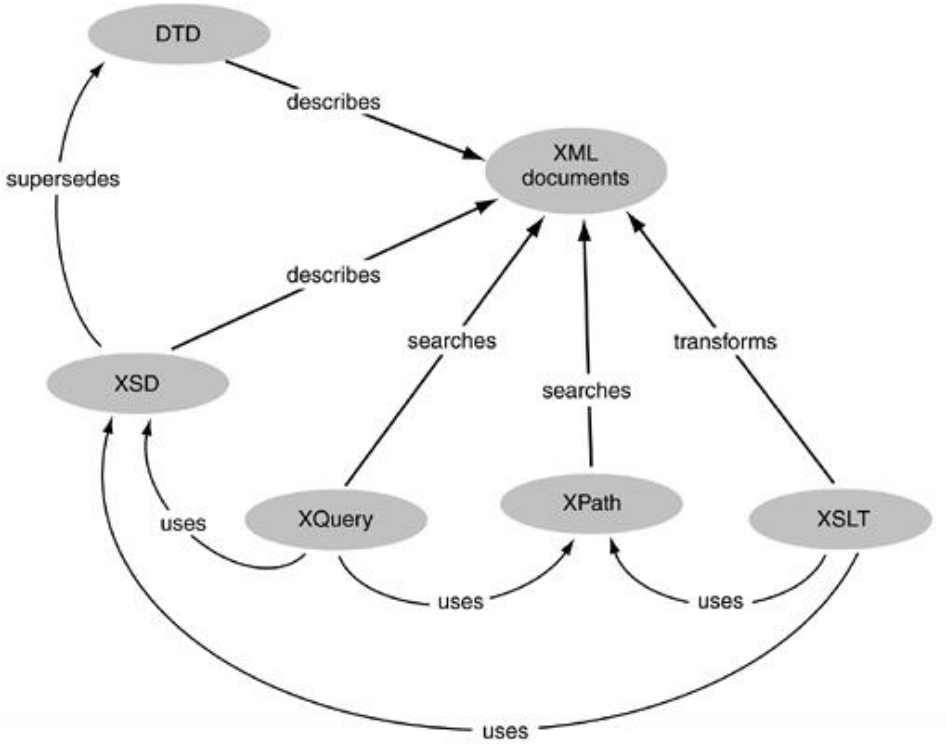


Figure 3.4 Relationship between XML specifications (Erl, 2004)

3.5.1 XML documents

An XML document has a logical and a physical structure. The physical structure consists of entities that are ordered hierarchically. The logical structure is explicitly described by markups that comprise declarations, elements, comments, character references, and processing instructions. XML documents that conform to the rules of XML mark-up are called "well-formed"; for example, each document must have a single top-level (root) element, and al1 tags must be correctly nested. A number of additional instructions arc permitted, such as comments, processing instructions, unparsed character data and entity references. Tags can also contain attributes in the form of name and value pairs, with the values enclosed in quotation marks. Figure 3.5 shows an example XML document.

```
<?xml version= “1.0” ?>
<! DOCTYPE student SYSTEM “student.dtd”>
<student category=“University”>
        <nationality>Serbia</nationality >
        <age>20</age>
</student>
```

Figure 3.5 An example XML document

Essentially, XML documents can be associated with and validated against a schema specification in terms of a Document Type Definition (DTD) or by using the more powerful XML Schema language.

3.5.2 Document Type Definition (DTD)

DTD defines the legal building blocks of an XML document. A DTD document contains the structural definition for the data in an XML document. It defines elements and attributes that can appear, default and fixed values for attributes and the relationships between elements. For example, figure 3.6 shows the example of DTD in an XML document. The student element has two child elements: nationality and age. Moreover, the student element was assigned an attribute called category, which has a validation rule limiting its possible value assignments to University or High-School.

```
<! DOCTYPE student [
<! ELEMENT student (nationality, age)>
        <! ATTLIST student CATEGORY (University | High-School)>
<! ELEMENT nationality (#PCDATA)>
<! ELEMENT age (#PCDATA)>
]>
```

Figure 3.6 The DTD of the XML document from figure 3.5

3.5.3 XML Schema Definition (XSD)

XSD is a comprehensive data modelling language for XML documents. Unlike DTDs, the XSD is an actual implementation of the XML language; schemas are themselves XML documents. XSD provides the structural and validation-related features offered by the DTD language within an extended feature set consisting of many more variations and options in how to model and establish validation criteria for XML document data.

The XML Schema document format is very flexible and highly extensible. One of the most important features introduced by the XML Schema specification is the wide range of support for data types and namespaces. In the figure 3.7, represent the XSD schema from the previous DTD in figure 3.6.

```
<?xml version= “1.0” ?>
<xsd:schema xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>
<xsd:element name=“student”>
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name=“nationality” type=“xsd:string”/>
                        <xsd:element name=“age” type=“xsd:int”/>
                </xsd:sequence>
                <xsd:attribute name= “category”>
                        <xsd:simpleType>
                                <xsd:restriction base= “xsd:string”>
                                        <xsd:enumeration value= “University”/>
                                        <xsd:enumeration value= “High-School”/>
                                <xsd:restriction>
                        </xsd:simpleType>
                </xsd:attribute>
        <xsd:complexType>
</xsd:element >
```

Figure 3.7 An example of XSD schema

3.5.4 XML Path (XPath)

XPath is a language used for finding data in XML documents by parsing those XML documents for specific values. XPath performs parsing of XML documents by applying an expression to the text of an XML document. The effective result is that an XPath expression allows navigation through XML document elements and attributes, retrieving items and values that match the expression passed into the XML document by XPath. Figure 3.8 shows the XPath expression that selects all student elements that have the nationality with the value 'Serbian'.

```
//student[nationality='Serbian']
```

Figure 3.8 An example of XPath expression

3.5.5 XQuery

XQuery is a standardized language that can be used to query XML documents just as SQL (Structured Query Language) is used to query relational databases. The XQuery is structured in FLWOR expression (pronounced as "flower expression") which is a form of for loop. The term FLWOR is an acronym for the keywords: for, let, where, order by, and return. The overview of these clauses is given as follows.

- **For**: Selects a sequence of nodes for iteration.
- **Let**: Binds values to variables.
- **Where**: Serves as a filter for the nodes.
- **Order By**: Values-based ordering of the nodes.
- **Return**: Determines what to return, and is evaluated once for every node.

Let us consider a FLWOR expression that returns the name of each student that has Serbian nationality. The FLWOR expression for the query is written as in Listing 3.2.

Listing 3.2 Example of XQuery with FLWOR expression

```
for $x in document("students.xml")//student
where $x/nationality = "Serbian"
return $x/name
```

The query iterates student elements, one at a time, found in students.xml, and checks whether the student element qualifies the conditions. If so, it returns the name of each qualifying student.

3.6 XML database

Being semi-structured data, there are two main approaches to storing XML documents. The first one is using an XML-enabled database such as a relational database, or an object-oriented database. The second one is using a native XML database.

3.6.1 XML-enabled database (XED)

XED is a relational database that transfers data between XML documents and relational tables. It retrieves data for maintaining the relational properties between tables and fields, rather than to model XML documents.

3.6.2 Native XML Database (NXD)

The native XML database stores XML data directly. The database engine accesses the XML data without performing any conversion. This is the main difference between an XML-enabled database and a native XML database. This direct access in a native XML database can reduce processing time and provide better performance.

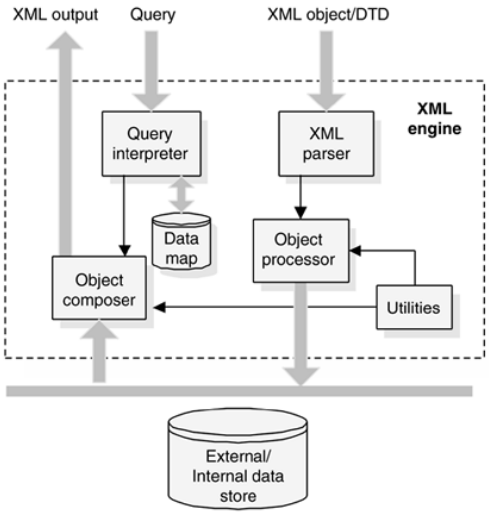


Figure 3.9 Architecture of Native XML Database (Fong, Wong, & Fong, 2021)

The diagram in figure 3.9 shows the storing and retrieving of XML data through the XML engine. The XML parser checks the syntactical correctness of the schema and ensures the incoming XML data objects are well formed. The object processor is used to store objects in the native XML store. The query interpreter resolves incoming requests and interacts with the object composer to retrieve XML objects according to the schemas defined by the administrator. Using the storage and retrieval schemas, the object composer constructs the information objects and returns them as XML documents.

3.7 eXist-db database

eXist-db is an open source native XML database system built for XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing. eXist-db supports many Web 2.0 technology standards, making it an excellent platform for developing web-based applications. Furthermore, eXist-db provides a pluggable module interface that allows extension modules to be easily developed in Java. These extension modules can provide additional XQuery functions through a custom namespace. The extension

modules have full access to the eXist db, its internal API, the context of executing XQuery and the HTTP session. The source code for extension modules should be placed in their own folder inside $EXIST_HOME/extensions/modules/src/org/exist/xquery/modules. The eXist-db architecture is shown in figure 3.10.

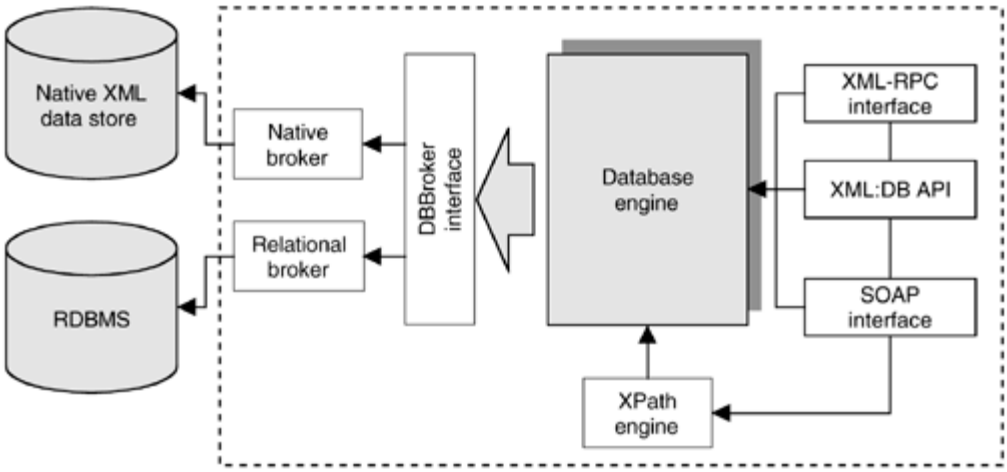


Figure 3.10 The eXist-db Architecture (Chaudhri, Rashid, & Zicari, 2003)

eXist-db provides an extension to XQuery for updating nodes in the database. Available update actions are: "insert", "delete", "replace", "value" and "rename". All update statements start with the keyword “update”, followed by an update action. The return type of the expression is empty(). For brevity, we explain only the update actions that have been used in the FXI system as follows.

- Update Insert

Syntax: update insert expr (into | following | preceding) exprSingle

Inserts the content sequence specified in expr into the element node passed via exprSingle. ExprSingle and expr should evaluate to a node set. If exprSingle contains more

than one element node, the modification will be applied to each of the nodes. The position of the insertion is determined by the keywords "into", "following" or "preceding":

into: The content is appended after the last child node of the specified elements.

following: The content is inserted immediately after the node specified in exprSingle.

preceding: The content is inserted before the node specified in exprSingle.

For Example, update insert <student><name>John</name> into //student[id="001"]

- Update Value

Syntax: update value expr with exprSingle

Updates the content of all nodes in expr with the items in exprSingle. If expr is an attribute or text node, its value will be set to the concatenated string value of all items in exprSingle. For example, update value //name[. = "John"] with 'Jim'

- Update Delete

Syntax: update delete expr

Removes all nodes in expr from their document. Expr cannot be the root element of a document. For example, update delete //student[id='Jim']

3.8 ANTLR (ANother Tool for Language Recognition)

ANTLR is a parser generator that uses LL(*) based recursive-descent parsers. It generates the source code for language recognizers, analyzers and translators from language specification (Parr, 2010). The latest version is 4.8 that supports Java, C#, C, Python and Ruby as target languages. Figure 3.11 presents the basic data flow for a translator. The lexer reads an

input character, divides it into tokens by using patterns that are specified in the grammar file and generates a token stream as output. The parser reads a token stream from the lexer and matches phrases in the language via the rules. After that parser generates Abstract Syntax Trees (AST) which can be processed with tree walker and generate the final tree representation as output.

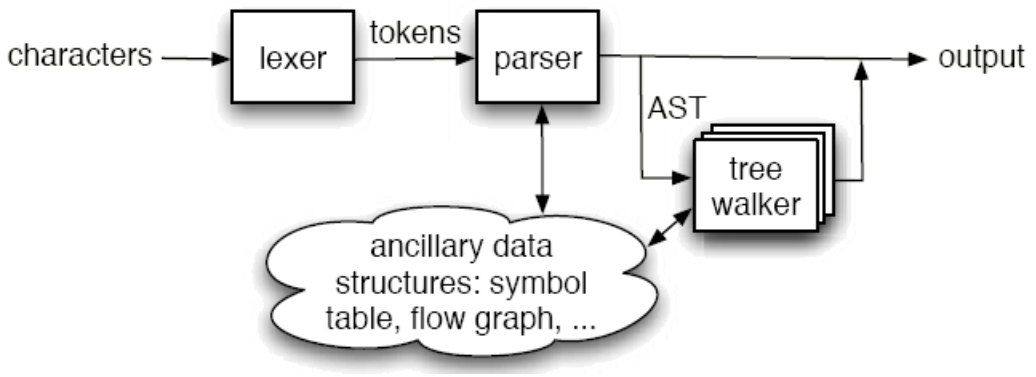


Figure 3.11 Overall translation data flow (Parr, 2007)

We create a grammar file (FuzzyXQuery.g) and run it with ANTLR tool. ANTLR automatically generates three files: Lexer.java, Parser.java and tokens file (as in figure 3.12). The important output from ANTLR is an AST. Here we will describe how to construct the AST.

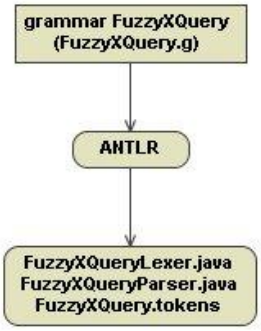


Figure 3.12 ANTLR generated Java files

First, we need to set the output option to AST in the grammar file (as shown in figure 3.13) because the default of grammar file does not create output in AST.

```
grammar FuzzyXQuery;
options {
        language = Java;
        output=AST;
        ASTLabelType=CommonTree;
}
```

Figure 3.13 Set options in the grammar file

Default AST construction will simply build a flat tree containing pointers to all the input token objects (as shown in figure 3.14). Therefore, we will add the AST construction operators and AST rewrite rules to facilitate handling the tree.

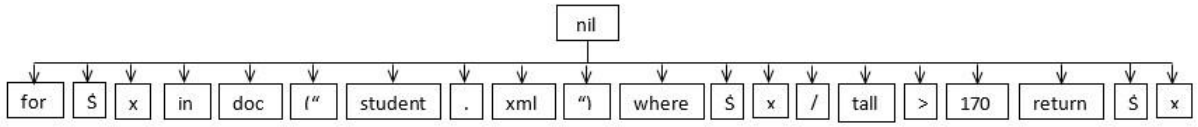


Figure 3.14 Default AST construction

I) Constructing ASTs Using Operators

There are 2 operators: ! and ^ symbol for constructing the ASTs. The meaning of these symbols is in table 3.5.

Table 3.5: Annotations for building AST nodes

Operator	Meaning
T!	discard T
T^	make T the root of this (sub) rule

Example:

comparisonexpr

: rangeexpr (generalcomp^ rangeexpr (priorityexpr!)?)? ;

From this rule, the generalcomp will be the root node of this rule and the priorityexpr will not be included in AST because of ! symbol. So, this rule will generate the AST as in figure 3.15.

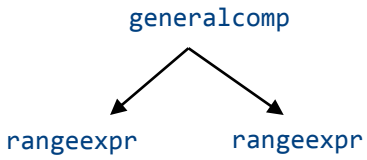


Figure 3.15 AST from comparisonexpr

II) Constructing ASTs with Rewrite Rules (Parr, 2007)

The rewrite syntax is more powerful than the operators. While the parser grammar specifies how to recognize input, the rewrites are generational grammars, specifying how to generate output. ANTLR figures out how to map input to output grammar as in figure 3.16.

```
rule    :       <<alt1>> -> <<build-this-from-alt1>>
        |       <<alt2>> -> <<build-this-from-alt2>>
        ...
        |       <<altN>> -> <<build-this-from-altN>>
        ;
```

Figure 3.16 Grammar for rewrite rules

Additionally, rewrite rules can add the imaginary node and omit the input elements.

- **Adding Imaginary node**

The imaginary token refers to tokens with a rewrite that are not found on the left of -> symbol. The imaginary node is used to group the unit chunk of nodes.

Example:

fuzzyexpr

:'#' QNAME -> FUZZY;

In this example, FUZZY is a node that is created from an imaginary token and used to group the fuzzyexpr.

- **Omitting input Elements**

Languages use many input symbols such as comma, semicolons, colons, parentheses, and so on, to indicate structure in the input. These symbols are not useful in the AST. Therefore, ANTLR uses the rewrite rule to delete these symbol tokens from the AST by omitting them from the rewrite specification.

Example:

whereclause

: 'where' exprsingle (thresholdexpr)? -> ^('where' exprsingle);

In this example, we omit the thresholdexpr from the whereclause tree

parenthesizedexpr

:'(' expr?')' -> expr?

In this example, we omit the parentheses -‘(’ and ‘)’ from the parenthesizedexpr.

3.9 Spring Boot

The Spring Framework has become the standard for building Java/Java EE-based enterprise applications because it offered a simpler approach, lightweight in terms of component code. However, Spring required a lot of explicit configuration in servlets and filters. In response to this problem, Spring Boot was proposed to take away boilerplate configuration. It is a new way to develop Spring applications with minimal friction from the framework itself. Auto-configuration eliminates much of the boilerplate configuration from traditional Spring applications. Spring Boot starters enable you to specify build dependencies by what they offer rather than explicit library names and versions (Walls, 2015). The features of Spring Boot are as follows (Spring Boot, 2020):

- Create stand-alone Spring applications
- Embedded Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' POMs to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

3.10 RESTful web services

REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client. It is the style that underlies the web as a

whole and has been used as a much simpler method than SOAP/WSDL for implementing web services (Sommerville, 2016).

A RESTful web service is identified by its URI (Universal Resource Identifier) and communicates using the HTML protocol. The HTTP methods are usually interpreted as follows:

1. GET is used to retrieve data for collection or the collection item.

2. POST is used to create a new resource.

3. PUT is used to update a resource.

4. DELETE is used to delete a resource.

The information exchanged by RESTful services is the resource representation in JSON (JavaScript Object Notation) format. For example, a student JSON data is {id: "001", name: "John", age: 26, height: 170}.

The REST interface is often preferred by developers and has become the standard for building services today because it has a lower overhead, simpler and is more efficient than traditional SOAP-based web services.

3.11 RESTful API in eXist-db

eXist-db is an open source native XML database system built for XML technology. It supports many Web 2.0 technology standards, making it an excellent platform for developing web-based applications. In the FXI system, the client-side accesses the eXist-db via REST Interface but server-side accesses by using XML:DB API.

The eXist-db provides a RESTful API through HTTP, which provides the simplest and quickest way to access the database. The basic operations defined by the eXist's REST API are

GET, POST, PUT and DELETE (existdb, 2014). When running in a servlet-context in our application, this servlet is configured to have a listen address at http://localhost:9999/exist/rest/. The server treats all HTTP request paths as paths to a database collection, i.e. all resources are read from the database instead of the file system. Relative paths are therefore resolved relative to the database root collection. For example, if you enter the following URL into your web browser: http://localhost:9999/exist/rest/db/data/studentdata.xml. The server will receive an HTTP GET request for the resource studentdata.xml in the collection /db/data in the database. The server will look for this collection and check if the resource is available and if so, retrieves its contents and sends them back to the client. To keep the interface simple, the basic database operations are directly mapped to HTTP request methods wherever possible. The following request methods are supported:

- GET

If the server receives an HTTP GET request, it first tries to locate known parameters. If no parameters are given or known, it will try to locate the collection or document specified in the URI database path, and returns a representation of this resource to the client. Note that when the located resource is XML, the returned content-type attribute value will be application/xml.

- PUT

Documents can be stored or updated using an HTTP PUT request. The request URI points to the location where the document will be stored. As defined by the HTTP specifications, an existing document at the specified path will be updated, i.e. removed, before storing the new resource. Also, any collections defined in the path that does not exist will be created automatically.

- DELETE

An HTTP DELETE removes a collection or resource from the database. For this, the server first checks if the request path points to an existing database collection or resource, and once found, removes it.

- POST

An HTTP POST request submits data in the form of an XML fragment in the content of the request that specifies the action to take.

Chapter 4

System Implementation

This chapter describes details of FXI implementation. Section 4.1 provides the fuzzy XQuery syntax in EBNF (Extended Backus Normal Form) notation. We present the representation of fuzzy data in section 4.2. In section 4.3 shows the use case diagram of the FXI system. The algorithm and fuzzy XQuery processing was given in section 4.4. Finally, we explain our system development and GUI (Graphical User Interface) in section 4.5.

4.1 Fuzzy XQuery EBNF grammar

The fuzzy XQuery syntax is described by using the EBNF notation as shown in figure 4.1 (Sae Ueng, Škrbić, Kansomkeat, & Apirada, 2017). We extended the standard XQuery in the *WhereClause* of the FLWOR (For-Let-Where-Order by-Return) statement with the Threshold, Priority, and Fuzzy Expression.

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

Figure 4.1 The snippet of fuzzy XQuery

A formal definition of these fuzzy constraints was given as follows:

I) A threshold Expression is an expression with the keyword *threshold* that we use to remove the results which have a membership degree less than the defined threshold from a result set. The threshold value is a real number between 0 and 1. If there is no threshold expression, the default value of threshold expression is 0.

II) A priority Expression is an expression with the keyword *priority* that specifies the importance of the corresponding constraints to the result. If the value of priority expression is higher, it means this constraint is more important. The priority value is also a real number between 0 and 1. If there is no priority expression, the default value of priority expression is 1.

III) A fuzzy expression is an expression that allows creation of a linguistic label or four types of fuzzy numbers in a query as follows:

- 'ling''('Qname')' indicates a linguistic label with the name given by Qname that was predefined in the system (see more in section 4.2).
- 'tri''('leftoffset','max','rightoffset')' indicates a Triangular fuzzy number with three arguments: left offset, maximum, and right offset.
- 'trap''('leftoffset','leftmax','rightmax','rightoffset')' indicates a Trapezoidal fuzzy number with four arguments: left offset, left maximum offset, right maximum offset, and right offset.
- 'interval''('leftoffset','rightoffset')' indicates an interval fuzzy number with two arguments: left offset and right offset.
- 'fs''('type','leftoffset','rightoffset')' indicates a fuzzy shoulder with three arguments: type of Fuzzy Shoulder (left shoulder or right shoulder), left offset, and right offset.

To illustrate the features of the fuzzy XQuery, we give four query examples. The first query (Listing 4.1) retrieves the students who are taller than 180 cm and their age is about 20 years old.

Listing 4.1 A fuzzy XQuery with fuzzy constants

```
for $x in document(student.xml)
where $x/height>180 AND $x/age= #fs(0,20,25)#
return $x
```

The # symbol is chosen to mark fuzzy constants. If we defined a linguistic label "young" that has value fs(0,20,25), the previous query could be simplified (Listing 4.2)

Listing 4.2 A fuzzy XQuery with a linguistic label "young"

```
for $x in document(student.xml)
where $x/height>180 AND $x/age= #ling(young)#
return $x
```

Queries can be enriched with additional constraints. The next query (Listing 4.3) contains the priority clause. The priority clause specifies the importance of the corresponding constraint to the overall result. If the value of the priority clause is higher, it means that the constraint has higher importance.

Listing 4.3 A fuzzy XQuery with the priority clauses

```
for $x in document(student.xml)
where $x/height>#tri(170,180,190)# priority 0.4 AND $x/age= #ling(young)# priority 0.6
return $x
```

Moreover, the query can include the threshold clause that limits the results and removes tuples with the global constraint satisfaction degree smaller than the value of threshold clause. In this example (Listing 4.4), the result set does not include the tuples with the global constraint satisfaction degree smaller than 0.2.

Listing 4.4 A fuzzy XQuery with a threshold clause

```
for $x in document(student.xml)
where $x/height>#tri(170,180,190)# priority 0.4 AND $x/age= #ling(young)# priority 0.6
threshold 0.2
return $x
```

4.2 Representation of fuzzy data in an XML document

We allow users to define fuzzy data representing imprecise data in an XML document by using linguistic labels. The linguistic labels (or linguistic terms) are used to represent expressions of a natural language (such as "young age," or "tall student") with associated degrees of membership. Our system supported four types of distributions of fuzzy numbers: triangular, trapezoidal, interval, and fuzzy shoulder (left shoulder and right shoulder).

To better understanding how the fuzzy data are presented, let us consider the following example (as shown in Figure 4.2). It represents a linguistic variable "age" containing a value

of linguistic label “young” that is corresponding to the fuzzy shoulder distribution (left shoulder) with the upper offset and lower offset of 20 and 25, respectively. Figure 4.3 shows the XML Schema used to define the linguistic label.

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

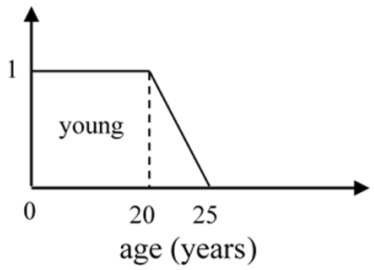


Figure 4.2 The definition of linguistic variable “age”

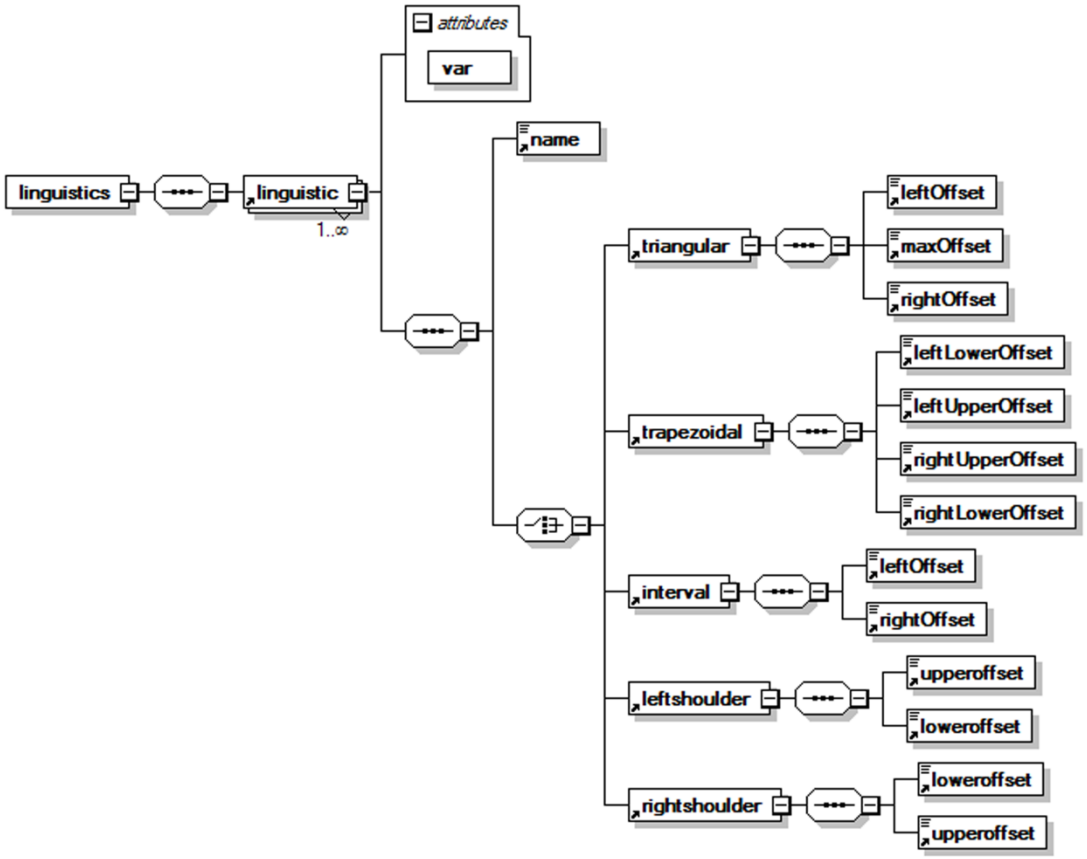


Figure 4.3 XML Schema for linguistic variable definition

Moreover, users can define the value of each element as the fuzzy number in the database (see figure 4.4) with four different types of fuzzy numbers, the same as in the linguistic label, as follows:

1. Triangle fuzzy number: tri(leftOffset, maxOffset, rightOffset)

2. Trapezoidal fuzzy number: trap(leftLowerOffset, leftUpperOffset, rightUpperOffset, rightLowerOffset)

3. Interval fuzzy number: interval(leftOffset, rightOffset)

4. Fuzzy shoulder: fs(type of fuzzy shoulder, lowerOffset, upperOffset)

```
<?xml version= "1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>Alex</name>
        <GPA>2.8</GPA>
        <age>20</age>
        <height>tri(150,200,250)</height>
    </student>
    <student>
        <name>Joe</name>
        <GPA>interval(3,4)</GPA>
        <age>21</age>
        <height>180</height>
    </student>
    <student>
        <name>Jack</name>
        <GPA>2.5</GPA>
        <age>fs(0,20,30)</age>
        <height>175</height>
    </student>
    <student>
        <name>Tom</name>
        <GPA>2.75</GPA>
        <age>19</age>
        <height>trap(155,160,170,175)</height>
    </student>
</students>
```

Figure 4.4 An example to specify the fuzzy values in an XML document

4.3 Use case diagram

The Fuzzy XQuery Interpreter (FXI) system provides the following key features:

- A user can input the fuzzy XQuery queries with the priority, threshold and fuzzy expressions.

- A user can define the linguistic variables in the eXist-db database.
- The system administrator can add/update/delete data in the eXist-db database.
- The system returns the results to the user via a web browser.

The use case model of the FXI system contains three actors and four use cases as shown in figure 4.5.

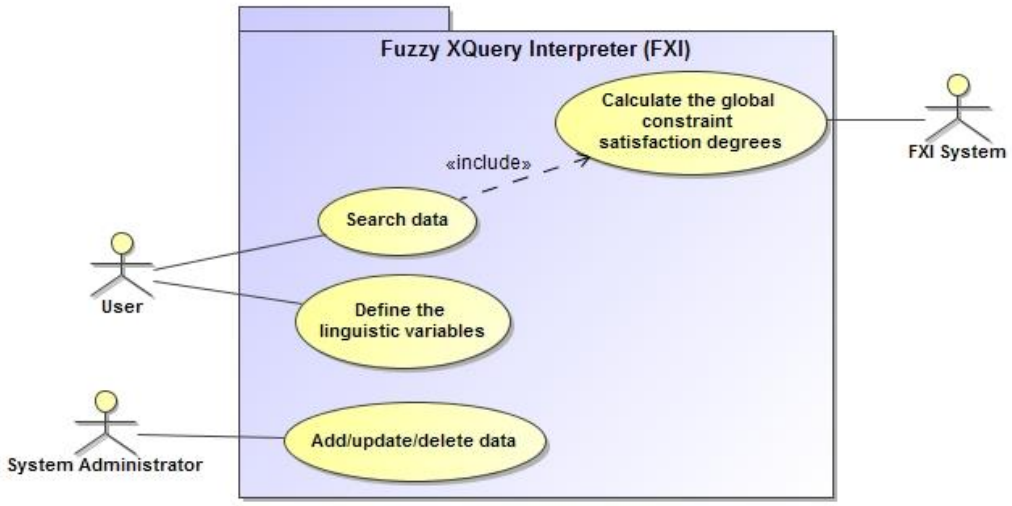


Figure 4.5 Use Case diagram of FXI system

The system contains the following actors:

- User: the main actor of this system.
- System Administrator: A role responsible for the smooth running of the system's technical resources (e.g. web server, database).
- FXI System: the system which is under consideration to accomplish a goal.

The description of use cases is shown in table 4.1, 4.2, 4.3 and 4.4 as follows.

Table 4.1 Tabular description of the "Search data" use case

Use Case Name:	Search data
Actors:	user
Description:	A user can search data in the database by inputting the fuzzy XQuery into the system

Preconditions:	The user knows the fuzzy XQuery syntax.
Postconditions:	The user gets the results from the database

Table 4.2 Tabular description of the "Calculate the global constraint satisfaction degrees" use case

Use Case Name:	Calculate the global constraint satisfaction degrees
Actors:	FXI system
Description:	Calculate the global constraint satisfaction degrees by using the GPFCSP concept
Preconditions:	There is a fuzzy XQuery
Postconditions:	Return the global constraint satisfaction degrees

Table 4.3 Tabular description of the "Define the linguistic variables" use case

Use Case Name:	Define the linguistic variables
Actors:	user
Description:	Create predefined fuzzy variables which can be used in queries by uploading an XML file or inputting in a form.
Preconditions:	The user knows the DTD of XML file for defining the variables
Postconditions:	Confirmation that the variables have been defined when finish.

Table 4.4 Tabular description of the "Add/update/delete data" use case

Use Case Name:	Add/update/delete data
Actors:	System administrator
Description:	Use to add, update or delete data in the database
Preconditions:	-
Postconditions:	Confirmation that data has been added/updated/deleted in the database

4.4 Fuzzy XQuery query processing

Mechanism of fuzzy XQuery query execution is shown in figure 4.6 while the algorithm that we use to calculate the global constraint satisfaction degree is shown in figure 4.7.

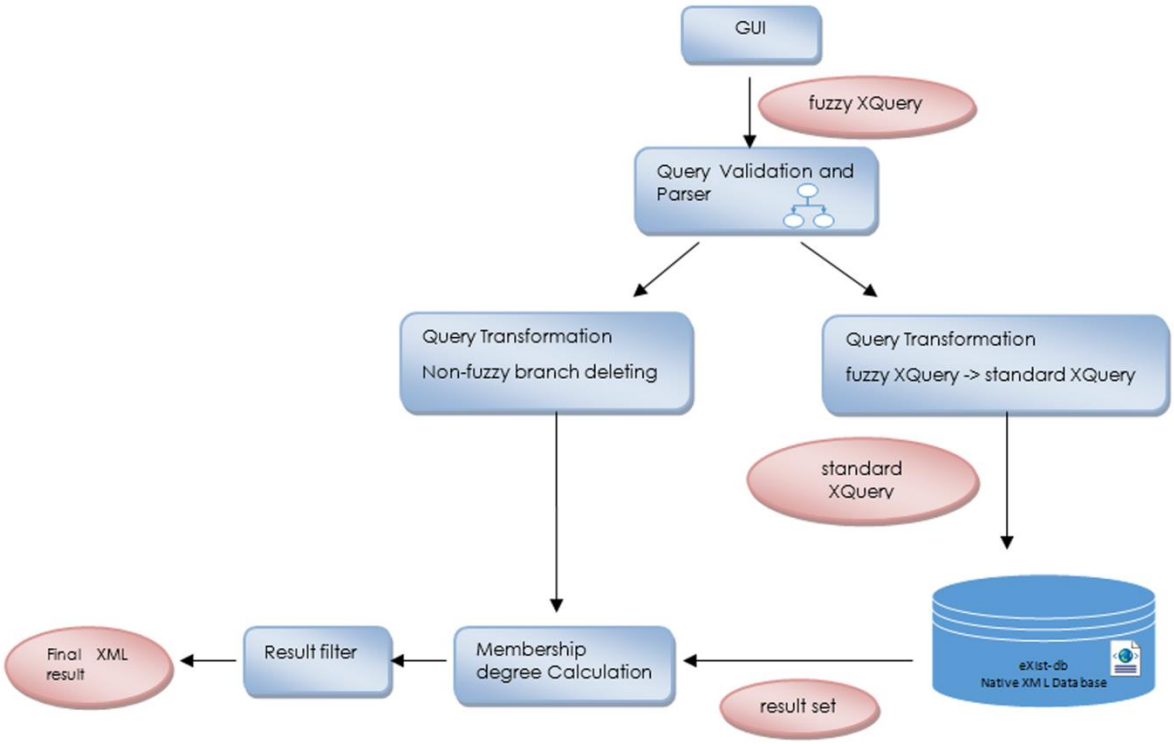


Figure 4.6 The process of Fuzzy XQuery execution

Now let us explain all details. When a user inputs a fuzzy XQuery query via GUI, the query is sent to the system, where it goes through multiple steps before returning the result set. First of all, the query syntax is validated against the given EBNF grammar (as can be seen in figure 4.1) and parsed into an Abstract Syntax Tree (AST) by using ANTLR (ANother Tool for Language Recognition). After that, the query is transformed in two ways. The first way, the query is transformed to a standard XQuery query by eliminating the fuzzy constraint from the fuzzy XQuery query. Then the system sends the standard XQuery to XML database. When the database returns the result set, they are again interpreted for calculating the global constraint satisfaction degree in *Membership degree Calculation*. The second way, the query is transformed by deleting the crisp constraints and sent to the *Membership degree Calculation*. Thereafter, the system goes into calculating the membership degrees by using GPFCSP concept. Once the fuzzy XQuery query has been evaluated, the results are filtered by the threshold value to produce the final results.

From the algorithm (figure 4.7), when the visited node is the conjunction 'AND', the system calculates the global constraint satisfaction degree (α) by calling the Łukasiewicz triangular norm (T_L) function. The Łukasiewicz triangular conorm (S_L) is used if the visited node is the disjunction 'OR'. If the fuzzy XQuery has priority expression, the system uses the triangular product conorm (S_P) to aggregate with the priority value. All is in accordance with the GPFCSP concept. Consider the comparison operator, two cases are possible: 1) "equality" (=) and "inequality" (!=) comparison of fuzzy data is calculated by calling the fuzzy compatibility operation (see section 3.3). In the case of 2) "Greater than" (>) and "less than" (<) comparison of fuzzy data is performed by calling the fuzzy ordering operation (see section 3.4).

Input: a fuzzy XQuery tree and a record of data

Output: a global constraint satisfaction degree (α)

1. Traverse the fuzzy XQuery tree
2. Find the *whereclause* node
3. Traverse the *whereclause* branch until end

   3.1 visited node is 'AND' then $\alpha \leftarrow T_L$(LeftBranch,RightBranch)

   3.2 visited node is 'OR', then $\alpha \leftarrow S_L$(LeftBranch,RightBranch)

   3.3 visited node is a comparison operator ('=', '!=', '>', '<'), then walk the child node

      3.3.1 If the current node is a comparison operator ('=', '!=', '>', '<'), then walk the child node

         (a) If the child node is a fuzzy constant **'ling'** and the right-hand side value is crisp value then α is the value of corresponding membership function in the given point of the linguistic variable

         (b) if the operator is '=' or '!=' then $\alpha \leftarrow$ Compatibility value

         (c) if the operator is '>' or '<' then $\alpha \leftarrow$ Fuzzy ordering value

   3.4 If there is a priority expression, then $\alpha \leftarrow S_P(\alpha$, 1-priority value)
4. Return α

Figure 4.7 The algorithm used to calculate the global constraint satisfaction degree

Now 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 4.5 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 0.5 meaning that we want the results that have the global constraint satisfaction degree more than 0.5.

Listing 4.5 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 4.6.

Listing 4.6 Transformation of the fuzzy XQuery query 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 query to the database. Third, we send the results back to the interpreter to calculate the global constraint satisfaction degree by calling the *Membership degree Calculation*. 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 4.7.

Listing 4.7 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 E4.

$$\alpha = T_L(S_P(\mu_{R_1^f}(v), 1-\rho(R_1^f)), S_P(\mu_{R_2^f}(v), 1-\rho(R_2^f))) \quad \text{(E4)}$$

In the equation E4, R_1^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 4.8 and the result set from the standard XQuery is shown in Listing 4.9. It is noticeable that Ana's GPA is not greater than 2.75. Consequently, the result in Listing 4.9 does not show Ana's record.

Listing 4.8 The snippet of student data

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

Listing 4.9 The result set from standard XQuery in Listing 4.6

```
<?xml version= "1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>John</name>
        <GPA>3.5</GPA>
        <age>25</age>
        <height>170</height>
    </student>
    <student>
```

```
        <name>Peter</name>
        <GPA>3.0</GPA>
        <age>21</age>
        <height>165</height>
    </student>
    <student>
        <name>Alex </name>
        <GPA>2.8</GPA>
        <age>20</age>
        <height>tri(150,200,250)</height>
    </student>
</students>
```

We calculate the constraint satisfaction degree(μ_{R_i}) for every constraint and every student as in Table 4.5. 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 has the left fuzzy shoulder which can be seen in Figure 4.8. 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 E4, we obtain the following:

$$\alpha_{John} = T_L(S_P(0,1-0.6), S_P(0.5,1-0.3)) \quad \text{(E5)}$$

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

$$\alpha_{John} = T_L(S_P(0,0.4), S_P(0.5,0.7)) = T_L(0.4,0.85) = 0.25 \quad \text{(E6)}$$

Table 4.5 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

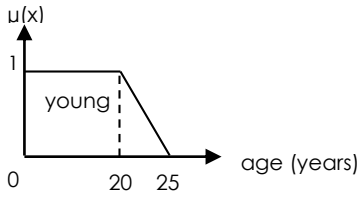


Figure 4.8 Membership function of *young*

The other students are calculated in the same way and are given in Table 4.6.

Table 4.6 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 4.10.

Listing 4.10 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>
```

4.5 System Development

The system architecture is shown in figure 4.9 (Sae-Ueng & Skrbic, 2020). There are three nodes: Database, Client and Web server under the Model-View-Controller (MVC) architecture. The database server is eXist-db, which is the native-XML database capable of executing XQuery queries. We use XML:DB API to connect to the database server. The client browser is connected to a server-side resource via RESTful API, and directly connects to the database via the eXist-db's RESTful API. The Apache Tomcat web server is based on the Java environment and Spring Boot Framework. We developed four main components: Calculation,

Fuzzy, ANTLRGrammar, and SubmitService. The grammar of the fuzzy XQuery is defined by using the ANother Tool for Language Recognition (ANTLR) that automatically generates the lexical analyzer and parser. The list of software used is shown in table 4.7. The detail of backend and frontend development is as follows:

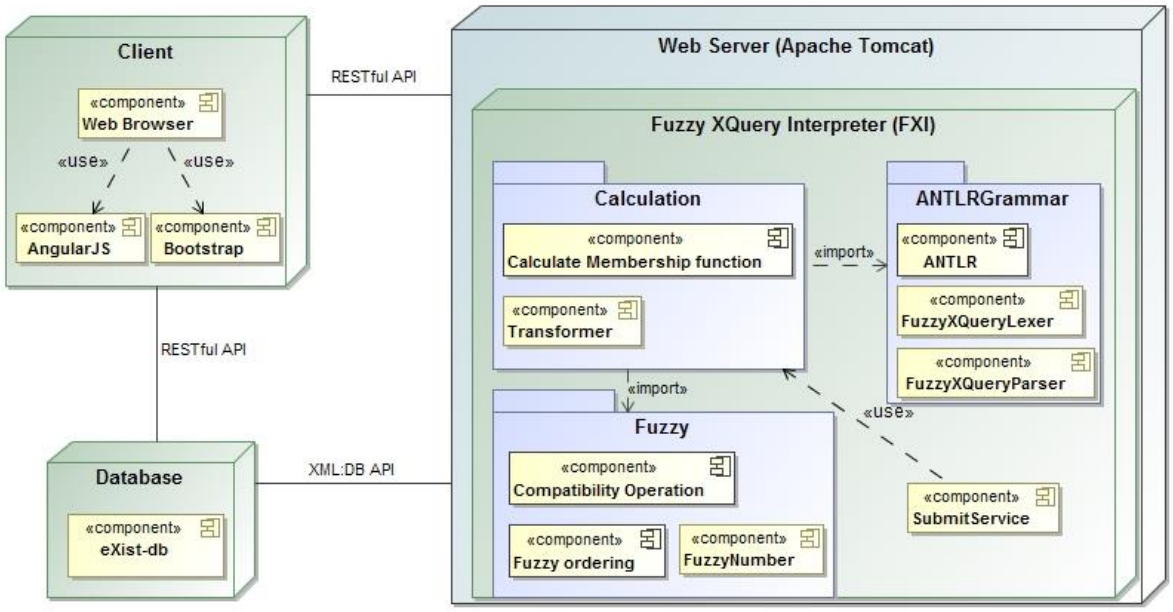


Figure 4.9 The system architecture of FXI system

Table 4.7 List of software

	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
	Spring Tool Suite (STS) IDE	3.7.1.RELEASE
	ANTLR	3.4 (Complete ANTLR 3.4 Java binaries jar)
	AngularJS	1.4.8

4.5.1 The Backend development

We developed the backend using Java programming language and based on Spring Framework. The main class diagram is shown in figure 4.10. There are five main packages as follows:

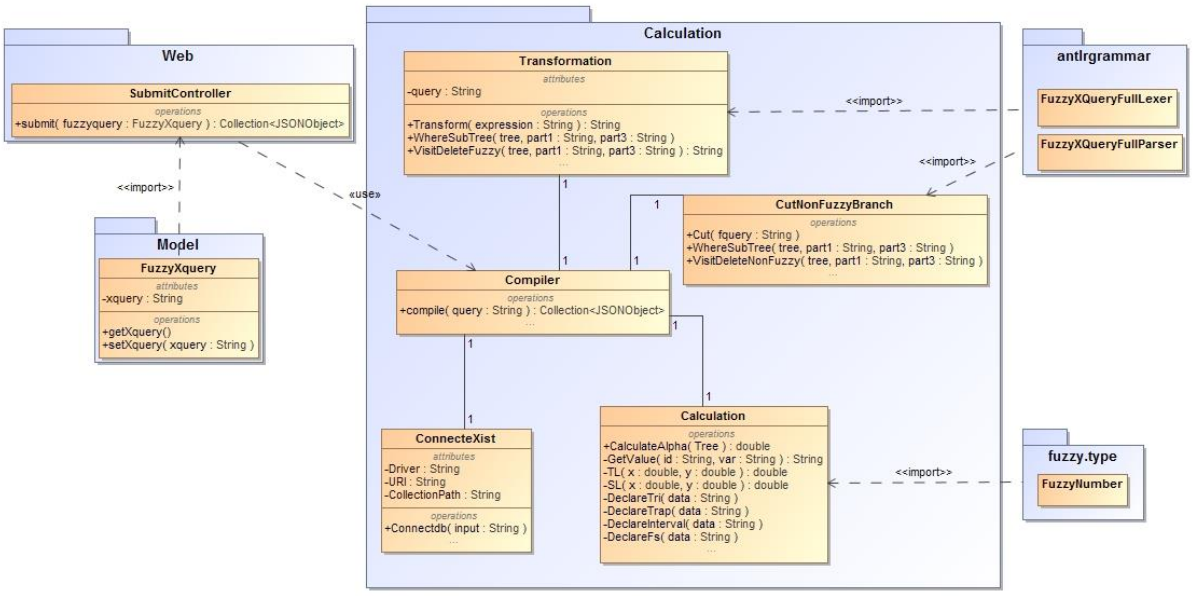


Figure 4.10 The main class diagram of FXI system

A. Antlrgrammar package

The fuzzy XQuery grammar was defined from the standard XQuery by using the EBNF 1.0 notation as in figure 4.1 (for the complete grammar, see Appendix A). We used the ANTLR (ANother Tool for Language Recognition) to parse the fuzzy XQuery and generate a lexical (FuzzyXQueryFullLexer.java) and a parser (FuzzyXQueryFullParser.java) class that is contained in this package.

B. Web package

In this package, we create the resource controller in a *SubmitController* class. Building RESTful web services and HTTP requests in Spring framework are handled by a controller. These components are easily identified by the *@RestController* annotation. The *SubmitController* class handles POST requests with *a FuzzyXquery* object for */submit* by

returning a JSON object of the student data's result set. The source code can be illustrated by the following:

```
@RestController
public class SubmitController {
    @RequestMapping(value ="/submit",method=RequestMethod.POST)
    public Collection<JSONObject> submit(@RequestBody FuzzyXquery fuzzyxquery)
{
        //call compiler class
    Compiler result = new Compiler();
    Collection<JSONObject> output =result.compile(fuzzyxquery.getXquery());
    return output;
     }
```

The *@RestController* annotation marks the class as a controller and converts the controller into a RESTful service that users can then access this resource from AngularJS application. The *@RequestMapping* annotation ensures that HTTP requests to */submit* are mapped to the submit() method. In this service, after the system gets the value *of FuzzyXquery* object, it will call the *compile* method of *Compiler* class to execute the fuzzy XQuery query.

C. Calculation package

This package consists of five classes (which are shown below) and used to execute the fuzzy XQuery. Now let us explain in detail how this package executes the fuzzy XQuery. When the *submit* service calls the *Compiler* class and passes a fuzzy XQuery in the URI, 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, creating an Abstract Syntax Tree (AST), and extracting 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 that have the fuzzy membership degree under the threshold value. Finally, the system returns the result set in a JSON object

C.1 Compiler class

This class is the main class for calling other classes, which has *compile* method that is the operation for receiving the fuzzy XQuery and returning the response to user interface.

C.2 Transformation class

This class is used to transform a fuzzy XQuery to a standard XQuery. There are three steps in this process**:** 1**)** check syntax and create an AST, 2**)** Delete fuzzy nodes and 3**)** inorder walk**.**

Step 1: Check syntax and create an AST

The system checks and validates its 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”

Thus, we will have the tree as in figure 4.11**.** It is noticeable that the *fuzzyexpr* has the FUZZY token in the branch**.**

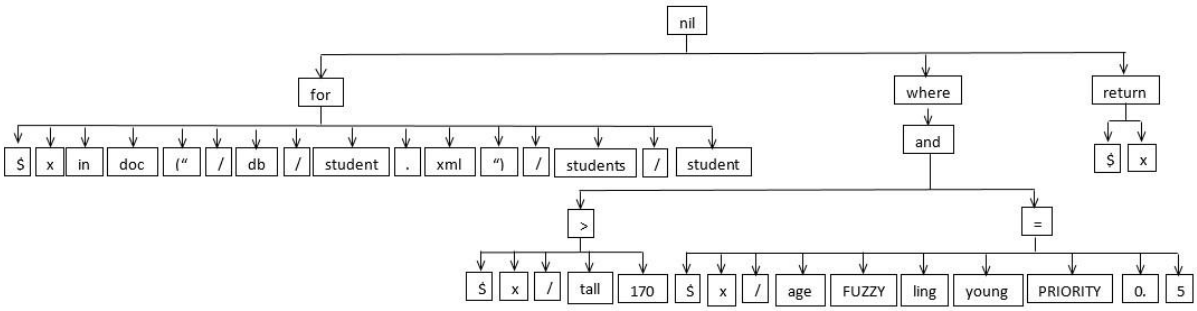


Figure 4.11 The AST created by ANTLR

Step 2: Tree traversal used for deleting fuzzy nodes in *Whereclause*.

We traverse the tree, extract only a *whereclause* subtree **(**as in figure 4.12**)** and delete the FUZZY node from the query**.**

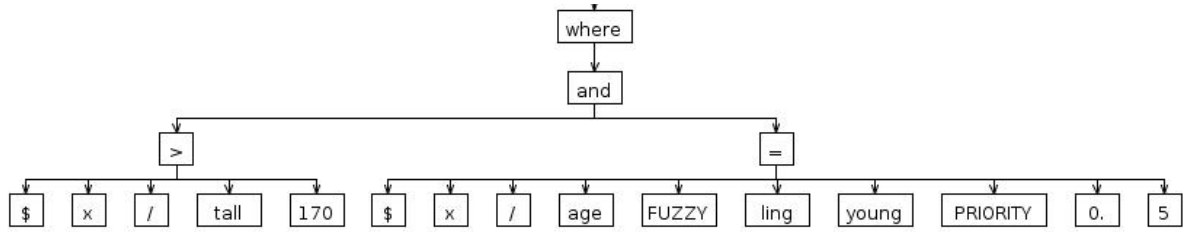


Figure 4.12 The *whereclause* subtree

The algorithm used to delete the FUZZY node from the *whereclause* subtree **(**as shown in figure 4.13**)** has three steps as follows**:**

1**.** Search FUZZY tokens from the child nodes**.** If found, delete the branch of the tree that has the FUZZY token**.**

2**.** If the parent of the branch **(**which was deleted in step 1**)** is a conjunction token **(**AND or OR**)**, delete the conjunction token and put the sibling branch instead of the conjunction token**.**

3**.** Traverse until FUZZY tokens are not found in the tree**.**

Finally, we have the new *whereclause* subtree as shown in figure 4.14**.**

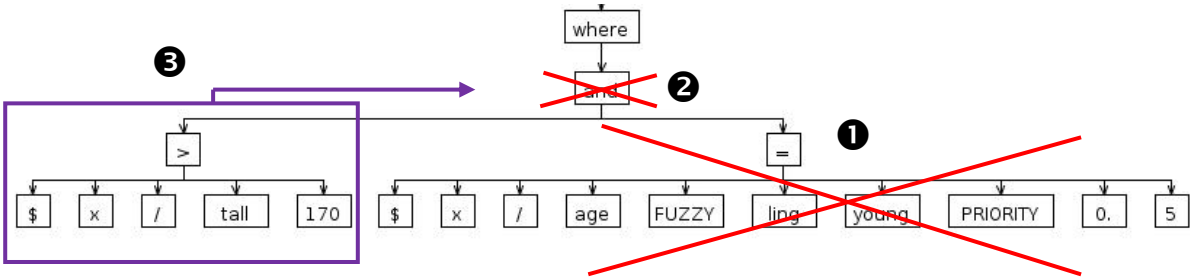


Figure 4.13 The algorithm used to delete the FUZZY node from *whereclause* subtree

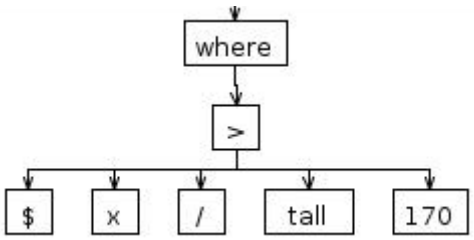


Figure 4.14 The *whereclause* subtree after the FUZZY token was deleted

Step 3: Inorder walk in the *Whereclause*.

After we have the new *whereclause* subtree, we walk through the *whereclause* subtree again and write it to a standard XQuery with inorder traversal**.** Inorder walk traverses the left subtree, visits the root node and finally traverses the right subtree**.** For example, in figure 4.15, we walk through the tree as follows**:**

1**.** Traverse the left subtree**:** $, x, /, tall

2**.** Visit the root node**:** >

3**.** Traverse the right subtree**:** 170

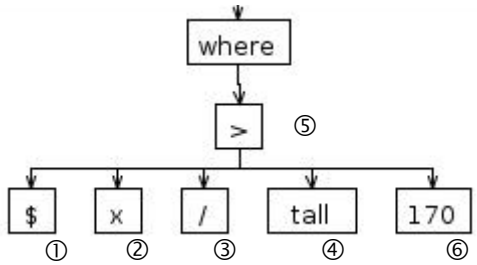


Figure 4.15 The inorder walk in *whereclaus*e subtree

Finally, from this example, we have the tree as shown in figure 4.16. That means we have the standard XQuery as follows**:**

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

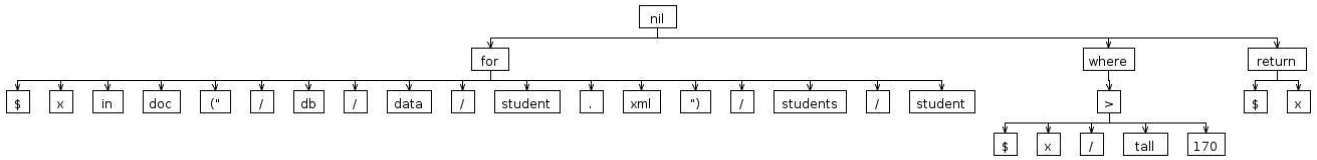


Figure 4.16 The tree after the fuzzy node was deleted

C.3 CutNonFuzzyBranch class

This class is used to delete the branches (in *whereclause* subtrees) that do not have the fuzzy nodes. Next, this subtree will send to the *Calculate* class for calculating the global constraint satisfaction degree.

C.4 ConnecteXist class

This class uses for connecting to the eXist-db database by *Connectdb* operation. We used the XML:DB API to connect the eXist-db because it is an API provides a common interface to native XML databases and supports the development of portable, reusable applications. This is the code for retrieving the data from eXist-db with XML:DB.

```
    //initialize database driver
    Class<?> cl = Class.forName(DRIVER);
    Database database = (Database) cl.newInstance();
    DatabaseManager.registerDatabase(database);

//get the collection
    Collection col = DatabaseManager.getCollection(URI + collectionPath);

//query a document

    System.out.println("Execute xQuery = " + input);

//Instantiate a XQuery service
    XQueryService service = (XQueryService) col.getService("XQueryService","1.0");
    service.setProperty("indent", "yes");

//Execute the query, print the result
    ResourceSet result = service.query(input);
    ResourceIterator i = result.getIterator();
while (i.hasMoreResources()) {
        Resource r = i.nextResource();
        System.out.println((String) r.getContent());
    }
```

In this example, the database driver class for eXist (org.exist.xmldb.DatabaseImpl) is first registered with the DatabaseManager. Next, we obtain a Collection object from the database manager by calling the static method DatabaseManager.getCollection(). The method expects a fully qualified URI for its parameter value, which identifies the desired collection. The URI should have the format like this:

xmldb:[DATABASE-ID]://[HOST-ADDRESS]/db/Collection

For instance, the URI like this: xmldb:exist://localhost:8080/exist/xmlrpc/db/data. The first part of the URI (xmldb:[DATABASE-ID]) determines which driver class to use. The database-id is used by the database manager to select the correct driver. This ID should always be "exist" if using eXist-db. The final part of the URI ([HOST-ADDRESS]/db/Collection) identifies the collection path, and optionally the host address of the database server on the network. eXist-db uses two different driver implementations: the first is a remote database engine using XML-RPC calls, the second is direct access to a local instance of eXist-db. The */db* is always the root collection.

C.5 Calculate class

The aim of this class is to walk the *whereclause* subtree of the fuzzy XQuery tree and calculate the global constraint satisfaction degree for every XML element in the result set by using the GPFCSP concept. Therefore, this class will calculate the degree by following the algorithm in figure 4.7.

D. Fuzzy Package

This package is used to define five types of fuzzy numbers as shown in figure 4.17 (Sukpisit, 2015):

1. The Triangular fuzzy number is represented with the FuzzyTriangle class.
2. The Trapezoidal fuzzy number is represented with the FuzzyTrapezoidal class.
3. The Fuzzy shoulder is represented with the FuzzyShoulder class.
4. The Fuzzy Interval is represented with the FuzzyInterval class.
5. The Crisp value is represented with the FuzzyCrisp class.

We use the Coordinate class, IntersectionArea class and AreaCalculator to calculate the compatibility operations and the fuzzy ordering operations.

E. Model package

This package has a *FuzzyXquery* class, which is a resource representation class that provides a plain old java object with field and two accessors (get and set) for the fuzzy XQuery data.

Figure 4.17 Class diagram of package fuzzy.type

4.5.2 The Frontend Development

We developed the Graphical User Interface (GUI) with AngularJS and Bootstrap. There are four main files: app.js, index.html, input.html, upload.html and managestudent.html as follows:

I) app.js

This file is a JavaScript file that contains an Angular module, a function config() and three controllers-SubmitController, UploadController, StudentController that controls three HTML pages: *input.html*, *upload.html* and *managestudent.html*, respectively.

i) Function config()

In *app.js* file has a function config which configures the routes by using the Angular UI Router (a third party routing module). The UI Router is a state-based approach. A state describes how each UI looks at a particular time. Each state consists of three components: url, templateUrl and controller.

- url: the URL of state
- templateUrl: the HTML template to be used
- controller: the controller which will be associated with the states.

In the FXI system, we have five states: home, about, upload, manageStudent and contact. The syntax to configure the routes is as follows:

```
app.config(function ($stateProvider, $urlRouterProvider) {
  $stateProvider
    .state('home', {
      url: '/home',
      templateUrl: '../html/input.html'
    })
    .state('about', {
      url: '/about',
      template: '../html/about.html'
    })
          .state('upload', {
                    url: '/upload',
                    templateUrl: '../html/upload.html'
          })
          .state('manageStudent', {
      url: '/manageStudent',
```

```
    templateUrl: '../html/managestudent.html'
  })
        .state('contact', {
    url: '/contact',
    template: '../html/contact.html'
  });

        $urlRouterProvider.otherwise("/home");

});
```

ii) SubmitController controls the *input.html* page. When a user submits the fuzzy XQuery query, the system will call the $http service, which is a core angular service that consumes web services via the browser's XMLHttpRequest object, with POST method (see figure 4.18) to send this XQuery to the "http://localhost:8080/submit" URL that is defined in the backend as the *SubmitService* RESTful service.

```
$http({
        method:'POST',
        url:'http://localhost:8080/submit',
        headers:{'Content-Type': 'application/json'},
        data:{xquery:$scope.xquery}
})
```

Figure 4.18 The snippet of source code for calling the $http service in *SubmitController*

iii) UploadController controls the *upload.html* page used for uploading an XML file to the database. We use the *ng-file-upload* of the Angular directive for uploading files to the eXist-db . This controller accesses to the database via the eXist-db RESTful API (see more in section 3.11) with the PUT method of $http service as can be seen in figure 4.19.

```
file.upload = Upload.upload({
  method:'PUT',
  url: 'http://localhost:9999/exist/rest/db/data/'+file.name,
  data: {file: file}
 });
```

Figure 4.19 The snippet of source code for calling the $http service in *UploadController*

iv) StudentController controls the *managestudent.html* page for adding, editing and deleting data in *studentdata.xml*. For getting the data, the controller accesses to the database via the eXist-db RESTful API with the GET method of $http service and uses *xml2json* component to convert the XML data to JSON format as you can see in figure 4.20.

```
$http.get('http://localhost:9999/exist/rest/db/data/studentdata.xml').then(function(response){
                var studentdata=x2js.xml_str2json(response.data);
                $scope.student = studentdata.students.student;
        }
```

Figure 4.20 The snippet of source code for getting data in *StudentController*

If we wish to add a new data, first we check if the studentID field already has the value being sent (check for duplicates). If it is does, the system returns the error to the user. However, if it does not, the controller accesses the database with the POST method of $http service and uses the *update insert* statement of XQuery update extension (see more in section 3.7) to insert the new data at the end of *studentdata.xml* as you can see in figure 4.21.

```
        var promise = $http({
        method:'POST',
        url: 'http://localhost:9999/exist/rest/db/data/studentdata.xml',
        headers: {'Authorization': 'Basic YWRtaW46MTIzNDU2','Content-
Type':'application/x-www-form-urlencoded'},
        data:'_query=update insert
<student><id>'+_id+'</id><name>'+_name+'</name><age>'+_age+'</age><gpa>'+_gpa+'</
gpa><height>'+_height+'</height></student> into //students'});
```

Figure 4.21 The snippet of source code for adding the new data in *StudentController*

For editing the data, the controller accesses to the database with the POST method of $http service and uses the *update value* statement to edit data as you can see in the snippet of source code in figure 4.22.

```
var promise = $http({
method:'POST',
url: 'http://localhost:9999/exist/rest/db/data/studentdata.xml',
headers: {'Authorization': 'Basic YWRtaW46MTIzNDU2','Content-Type':'application/x-www-form-urlencoded'},
data:'_query=update value //name[.="'+$scope.temp.name+'"] with "'+$scope.EditData.name+'"'});
```

Figure 4.22 The snippet of source code for editing the data in *StudentController*

For deleting the data, the controller accesses to the database with the POST method of $http service and uses the *update delete* statement to delete data as you can see in figure 4.23.

```
var promise = $http({
method:'POST',
url: 'http://localhost:9999/exist/rest/db/data/studentdata.xml',
headers: {'Authorization': 'Basic YWRtaW46MTIzNDU2','Content-Type':'application/x-www-form-urlencoded'},
data:'_query=update delete //student[id="'+data.id+'"]'});
```

Figure 4.23 The snippet of source code for deleting the data in *StudentController*

II) index.html

This is the main page of our application for calling the libraries and *app.js* to run.

III) input.html

This page uses to submit the fuzzy XQuery. When a user adds the fuzzy XQuery in the text box (as shown in figure 4.24) and click on the *Submit* button to submit the query into our system. After that the result will be shown as in figure 4.25. The table of results has six columns: ID, Name, Age, GPA, Height and degree. The first five columns are the data from *studentdata.xml* file in the database and the last column is the global constraint satisfaction degree of that record.

IV) upload.html

This page (as can be seen in figure 4.26) is used to upload an XML file for predefining the linguistic variables in eXist-db. We use *ng-file-upload* that is the lightweight Angular directive to upload files.

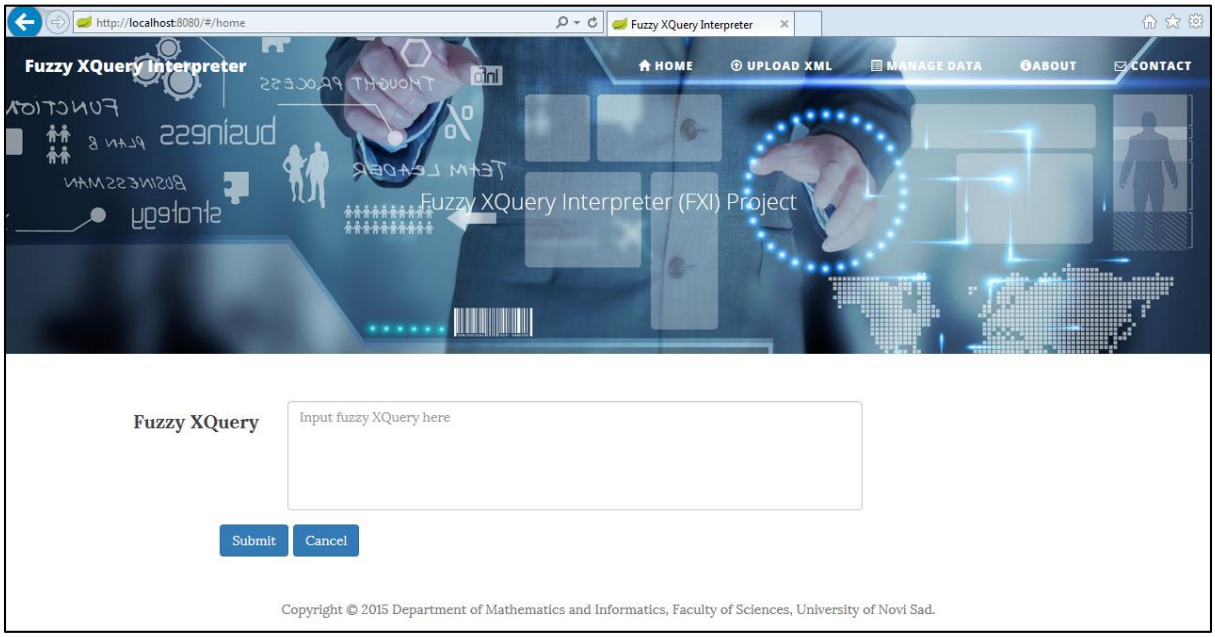


Figure 4.24 The *input.html* page

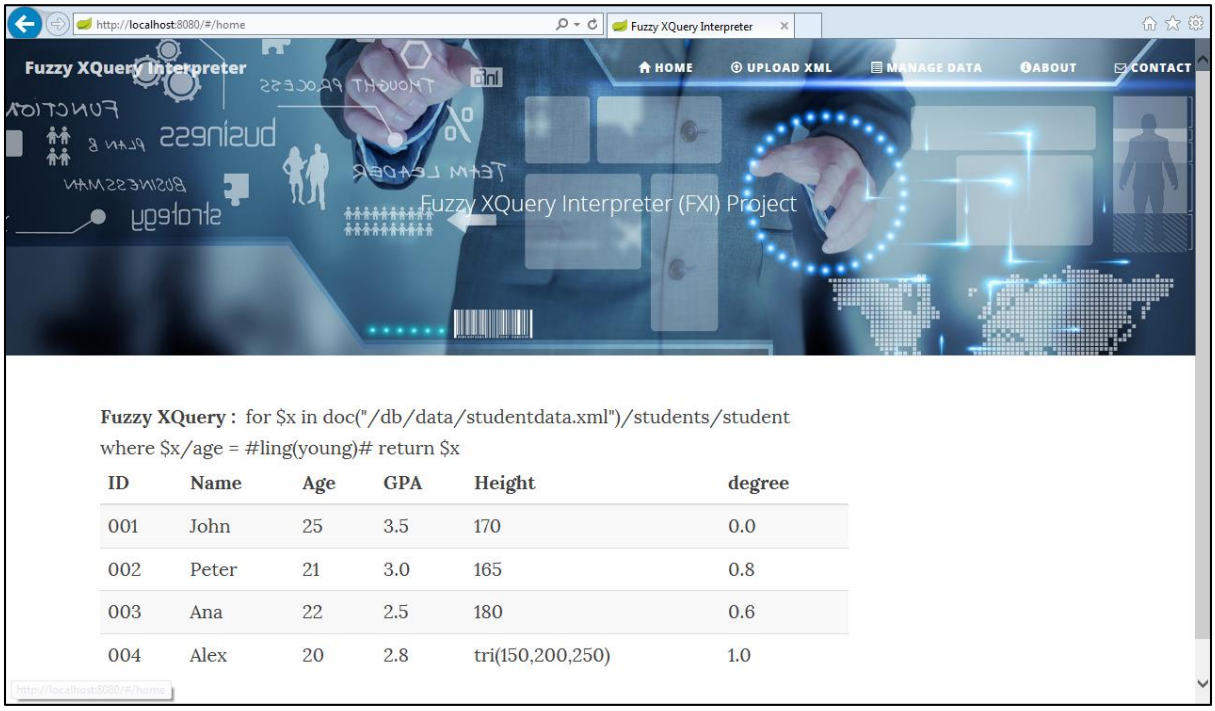


Figure 4.25 The *input.html* page with the result

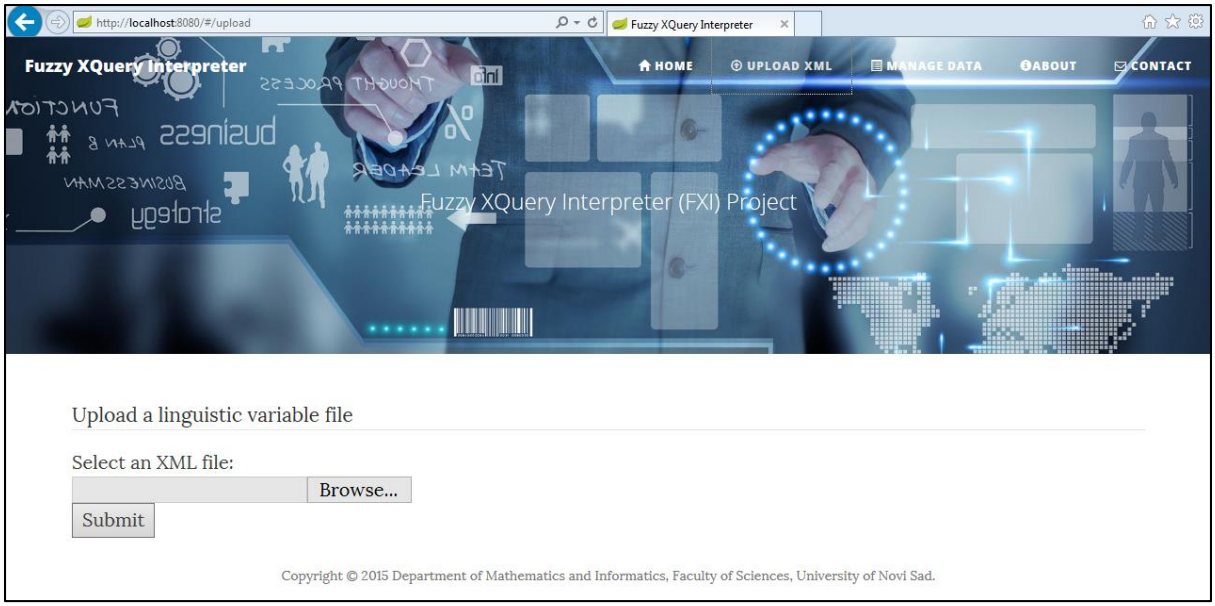


Figure 4.26 The *upload.html* page

V) managestudent.html

This page used to create, update, delete of student data in the *studentdata.xml* file in the eXist-db. When the browser loads the *div* element with ng-controller directive 'StudentController', the controller module in the *app.js* file executes. It makes the $http service to get all students initially and display in this HTML file as a table as shown in figure 4.27. If the user clicks the *New Data* button, it shows five text boxes to input the new data as in figure 4.28. When the user clicks the *Edit* button to edit data in any lines of the table, it will show the same five text boxes with the old data in that row as in figure 4.29. If the user clicks the *Delete* button, it will delete that record immediately.

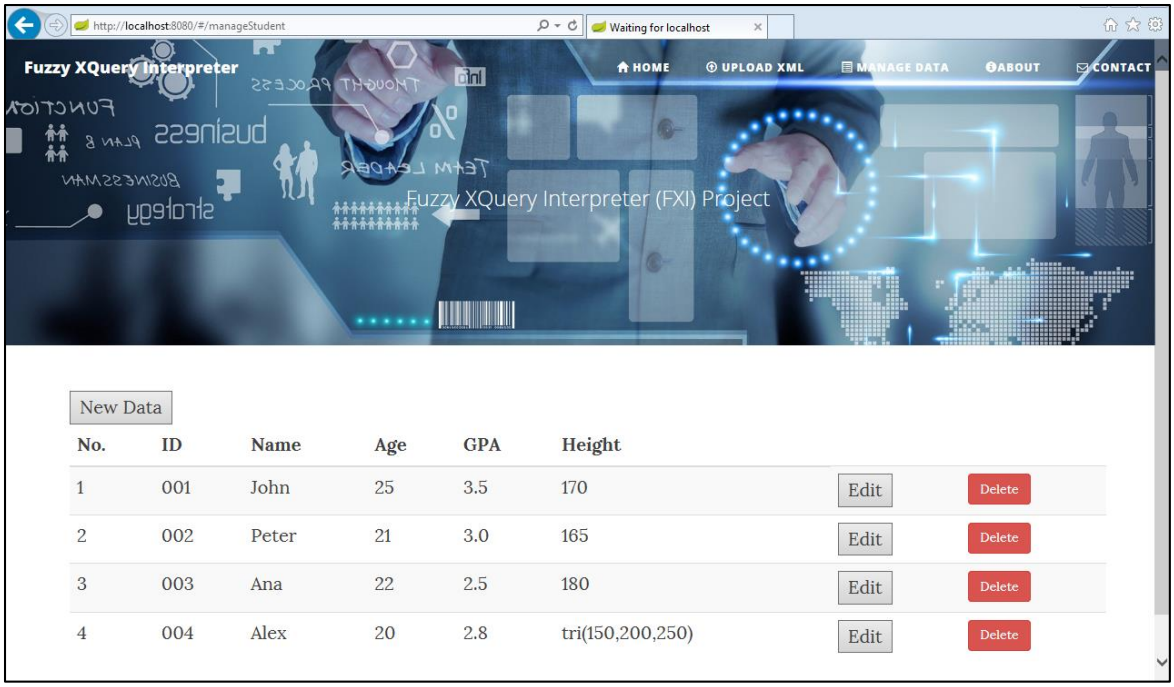


Figure 4.27 The *managestudent.html* page

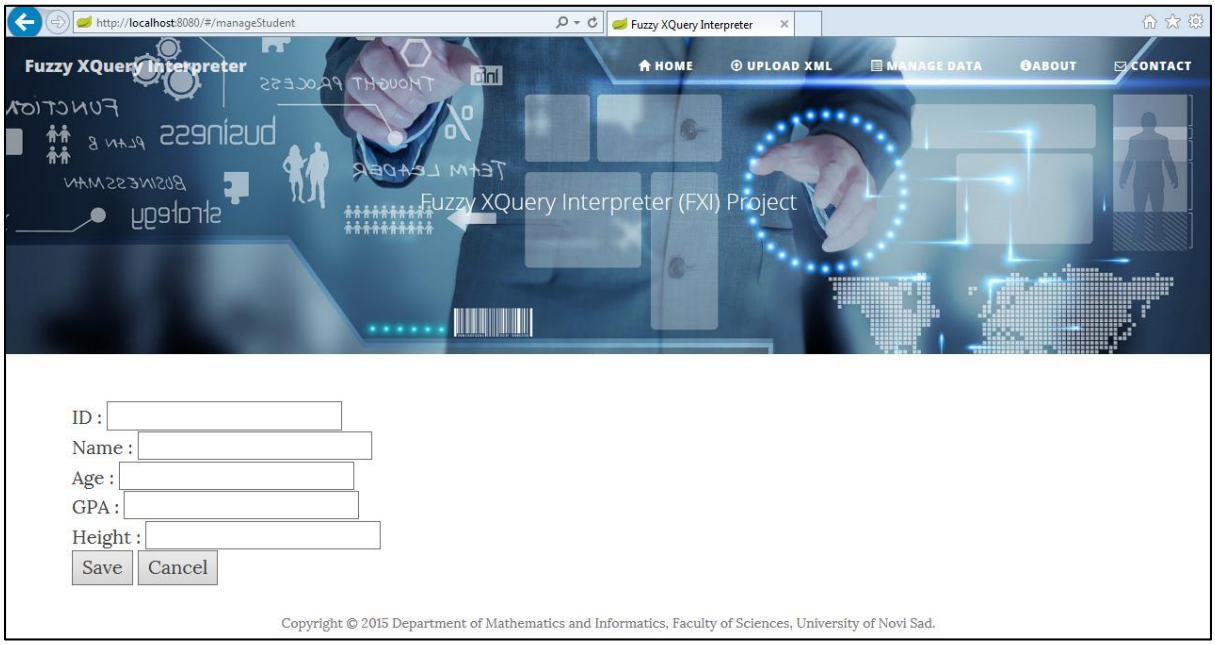


Figure 4.28 The *managestudent.html* page when a user wants to add new data

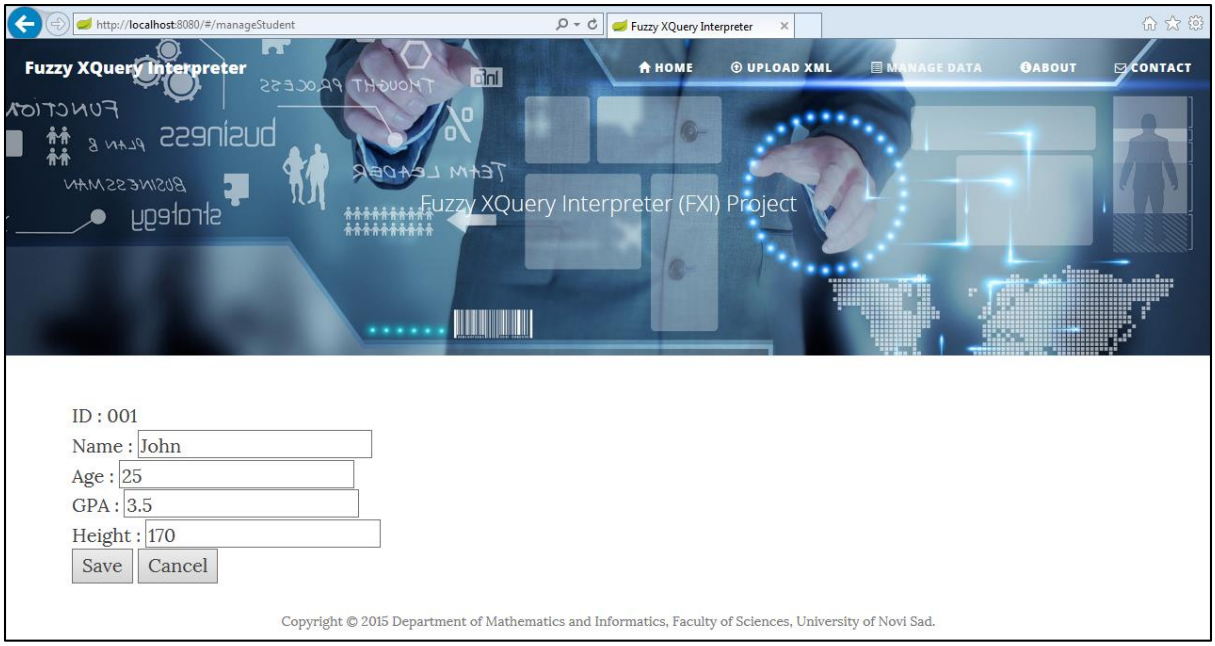


Figure 4.29 The *managestudent.html* page when a user wants to edit data

Chapter 5

System Testing

This describes in detail experiments conducted to evaluate the accuracy and performance of the FXI system. The objective is to determine how the FXI system meets the requirements, does it respond correctly to all kinds of inputs and does it perform its functions within an acceptable time frame.

5.1 Correctness Testing

The purpose of correctness testing is to prove that the FXI system can calculate the global constraint satisfaction degree correctly for all kinds of possible inputs. The structure of fuzzy XQuery can have the conjunction “AND” or “OR” as in figure 5.1.

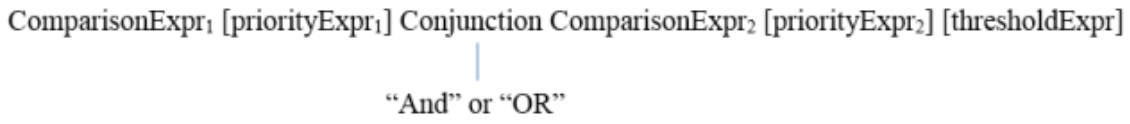


Figure 5.1 The conditional structure of fuzzy XQuery

Each condition (Comparison Expression) can have four relational operations (=,!=,>,<) as in figure 5.2. A Variable Expression (VariableExpr) can take values of crisp number type or four types of fuzzy numbers: triangle, trapezoidal, Interval and two kinds of fuzzy shoulders: left shoulder and right shoulder. In the same way, fuzzy expressions (FuzzyExpr) can be defined as linguistic variables or four types of fuzzy numbers.

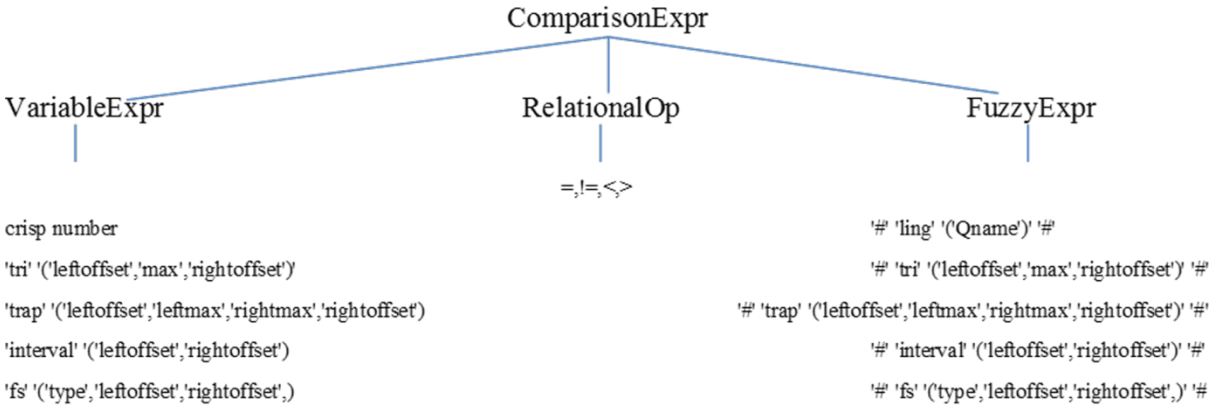


Figure 5.2 Possible values of ComparisonExpr

Therefore, we consider all possible input test cases as follows (see table 5.1):

First, we consider all types of fuzzy numbers in fuzzy Expression (FuzzyExpr) of the first and second conditions. For example, the Fuzzy Expression of the first condition has a triangle fuzzy number and the second condition has a trapezoidal fuzzy number as below:

Where $x/height=#tri(170,180,190)# AND $x/age=#trap(30,40,50,60)#

There are four possible fuzzy numbers in the first conditions and second conditions. However, the fuzzy shoulder has two types: left shoulder and right shoulder. Thus, we have 5x5=25 main test scenario as in table 5.2.

Second, the assignment of values to the variable expression (VariableExpr) can be defined as a crisp number or four types of a fuzzy number.

Third, it is possible to have a priority expression in a condition of fuzzy XQuery. Therefore, there are four possible cases of priority expression as follows:

A. The query does not have any priority expression.

B. The query has priority expressions in both conditions.

C. The query has a priority expression only in the first condition.

D. The query has a priority expression only in the second condition.

Fourth, we consider the relational operator as one of the four operators: =,!=,>, and < from both conditions.

Fifth, there are two conjunctions: AND or OR in a fuzzy XQuery.

Lastly, it is possible to have a threshold expression in the fuzzy XQuery.

Based on the above considerations, we have 22,400 test cases and have constructed the test case tables as shown in Appendix B. Table B.1 shows the various inputs of T2 test scenario with conjunction AND in the query and table B.2 describes the test data of variable expressions in the test cases.

Table 5.1 Possible input test cases

Factors					
Fuzzy Expression	**Variable Expression**	**Priority expression**	**Relational operator**	**Conjunction**	**Threshold expression**
Triangle	Crisp number	(A) No priority expression	=	AND	No threshold
Trapezoidal	Triangle	(B) Both conditions have priority expression	!=	OR	Has threshold
Interval	Trapezoidal	(C) Priority expression in the first condition	>		
Leftshoulder	Interval	(D) Priority expression in the second condition	<		
Rightshoulder	Leftshoulder				
	Rightshoulder				

Table 5.2 Test Scenarios

Test Scenario	First Fuzzy Expression	Second Fuzzy Expression
T1	Triangle	Triangle
T2	Triangle	Trapezoidal
T3	Triangle	Interval
T4	Triangle	Leftshoulder
T5	Triangle	Rightshoulder
T6	Trapezoidal	Triangle
T7	Trapezoidal	Trapezoidal
T8	Trapezoidal	Interval
T9	Trapezoidal	Leftshoulder
T10	Trapezoidal	Rightshoulder
T11	Interval	Triangle
T12	Interval	Trapezoidal
T13	Interval	Interval
T14	Interval	Leftshoulder
T15	Interval	Rightshoulder
T16	Leftshoulder	Triangle
T17	Leftshoulder	Trapezoidal
T18	Leftshoulder	Interval
T19	Leftshoulder	Leftshoulder
T20	Leftshoulder	Rightshoulder
T21	Rightshoulder	Triangle
T22	Rightshoulder	Trapezoidal
T23	Rightshoulder	Interval

Test Scenario	First Fuzzy Expression	Second Fuzzy Expression
T24	Rightshoulder	Leftshoulder
T25	Rightshoulder	Rightshoulder

5.2 Performance Testing

Performance testing is a testing practise performed to determine how a system performs in terms of responsiveness and stability under a particular workload. We analyze the performance of the FXI system based on a comparison of two data sets under a server containing the Intel Core2 Duo 2.4 GHz processor with 4 GB RAM memory running the 64-bit Windows 7 operating system. The data set was the student data from Prince of Songkla University about 5,000 records with five elements as in Listing 5.1.

Listing 5.1 The student data

```
<students>
  <student>
    <id>10001</id>
    <name>JIRAWAN</name>
    <age>25</age>
    <height>170</height>
    <gpa>3.23</gpa>
  </student>
</students>
```

The efficiency of the FXI system was evaluated by measuring how the query execution time varies depending on the number of fuzzy values in the query and data. There are two experiments. First, we executed a fuzzy XQuery with various fuzzy variables as shown in Listing 5.2 and 5.3. The result shows that the execution time increases with the increasing number of fuzzy variables, as expected (see Table 5.1).

Listing 5.2 The fuzzy XQuery with one fuzzy variable

```
for $x in doc(/apps/FXIdb/studentdata.xml)
where $x/age=#trap(18,20,22,25)#
and $x/height>160
return $x
```

Listing 5.3 The fuzzy XQuery with two fuzzy variables

```
for $x in doc(/apps/FXIdb/studentdata.xml)
where $x/age=#trap(18,20,22,25)#
and $x/height=#tri(100,150,200)#
return $x
```

Table 5.3 Fuzzy variable/Execution time

789 KB file size	1 variable	2 variables
Fuzzy XQuery	25 s	38 s

The second experiment used the same data as in the first, but we executed a query regarding the size of fuzzy data in an XML document. We ran the query in Listing 5.3 randomizing fuzzy values only in the *age* field in three cases: 1,000, 2,500 and 4,000 records, respectively. The result shows that the value type as a factor (crisp value or fuzzy values) does not make a significant difference in the response times (see Table 5.2).

Table 5.4 Fuzzy data/Execution time

	Crisp 5,000 records	Crisp 4,000 records, fuzzy data 1,000 records	Crisp 2,500 records, fuzzy data 2,500 records	Crisp 1,000 records, fuzzy data 4,000 records
Fuzzy XQuery	38s	42s	43s	44s

Chapter 6

Conclusion

This thesis proposes fuzzy XQuery expansions that extend the standard XQuery capabilities with priority, threshold and fuzzy expressions. The *priority* expression specifies the importance of the corresponding constraints to the result, while the *threshold* expression use to eliminate the results, which have a membership degree less than the defined threshold value from a result set. The fuzzy extensions of the XML database allowed the use of fuzzy values and predefined linguistic labels that could be used later in the queries. One of the main results is a detail of an algorithm that calculated the global constraint satisfaction degrees using Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) theory. The introduction of fuzzy values in XQuery queries raises the question of comparing fuzzy values, calculating compatibility, as well as aggregating the satisfaction degree of individual conditions into a global constraint satisfaction degree. The implementation of a software product in which these solutions are integrated is called Fuzzy XQuery Interpreter (FXI) – an interpreter of processing fuzzy XQuery queries with priorities based on open-source technologies and native XML open-source database.

Furthermore, the two types of experimental evaluations of results were performed – correctness testing and performance testing. The purpose of the correctness testing is to show that the FXI calculates the exact global satisfaction degrees, for different types of query constructs and overall available data 22,400 cases were tested experimentally, all with a successful outcome. The performance testing was done by measuring the variation of query execution time depending on the number of fuzzy constructions in the query and data. There were two experiments. First, queries were made with one and two fuzzy variables overall 5,000 elements of the data set. Second, the same data were used, but the dependence on the amount of fuzzy data in the XML document was measured by varying the fuzzy values.

The largest contribution of this thesis is the successful development of the fuzzy XQuery interpreter offered as a web application implemented using Java and eXist-db. There are various ways to support fuzziness in XQuery as described in chapter II. Fuzzy membership degree calculations within the product are based on GPFCSP, as in some previous approaches, however, this is an implementation written from scratch, and for the first time, it is not based on external commercial products used for evaluations, calculations and data storage. Instead,

we only use Java and open source native XML database written in Java. Everything else has been implemented by hand within the product. This includes an engine that implements fuzzy ordering based on Bodenhofer's order that allows the usage of relational operators (<, <=, >, >=) in queries, as well as an innovative engine, used to calculate compatibility degree between two fuzzy values. Every aspect of the implementation has been tested and validated for correctness and performance.

We plan to base our future work on applications of querying mechanisms in the Resource Description Framework (RDF). The RDF is a language used for representing metadata about web resources that can be vague or ambiguous. Thus, we believe that described methods could be used in the intelligent data representations related to web resources.

Bibliography

Almendros-Jiménez, J. M., Becerra-Terón, A., & Moreno, G. (2017). FSA-SPARQL: Fuzzy Queries in SPARQL.

Almendros-Jiménez, J., Tedesqui, A., & Moreno, G. (2015). Fuzzy xpath through fuzzy logic programming. *New Generation Computing, 33*(2), 173-209.

Alonso, S. K. (n.d.). *MEMBERSHIP FUNCTIONS*. Retrieved September 27, 2020, from eMathTeacher: Mamdani's Fuzzy Inference Method: http://www.dma.fi.upm.es/recursos/aplicaciones/logica_borrosa/web/fuzzy_inferencia/funpert_en.htm

Amer-Yahia, S., Lakshmanan, L., & Pandit, S. (2004). FleXPath: flexible structure and full-text querying for XML. *SIGMOD '04: the 2004 ACM SIGMOD international conference on Management of data* (pp. 83-94). Paris, France: ACM. doi:10.1145/1007568.1007581

Bodenhofer, U. (2008). Orderings of fuzzy sets based on fuzzy orderings. Part I: the basic approach. *Mathware & Soft Computing*, 201-218.

Campi, A., Damiani, E., Guinea, S., Marrara, S., Pasi, G., & Spoletini, P. (2009). A fuzzy extension of the XPath query language. *Journal of Intelligent Information Systems, 33*, 285-305. doi:10.1007/s10844-008-0066-3

Campi, A., Guinea, S., & Spoletini, P. (2014). An Operational Semantics for XML Fuzzy Queries. *the International Conference on Fuzzy Computation Theory and Applications (FCTA-2014)*, (pp. 205-210). Rome, Italy. doi:10.5220/0005155502050210

Chaudhri, A. B., Rashid, A., & Zicari, R. (2003). *XML Data Management: Native XML and XML-enabled Database Systems.* Addison Wesley.

Dubois, D., Fargier, H., & Prade, H. (1996). Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 287-309.

Erl, T. (2004). *Service-oriented architecture; a field guide to integrating XML and web services.* Prentice Hall PTR.

existdb. (2014). Retrieved September 27, 2020, from REST-Style Web API: http://exist-db.org/exist/apps/doc/devguide_rest.xml

Fong, J., Wong, H. K., & Fong, A. (2021). *Performance Analysis between an XML-Enabled Database and a Native XML Database.* Retrieved from http://etutorials.org/.

Fredrick, E., & Radhamani, G. (2010). A GUI Based Tool for Generating XQuery and Fuzzy. *International Journal of Computer Applications, 1*(17), 54-58.

Fredrick, E., & Radhamani, G. (2011). INFORMATION RETRIEVAL USING XQUERY PROCESSING TECHNIQUES. *International Journal of Database Management Systems (IJDMS), 3*(1), 50-58. doi:10.5121/ijdms.2011.3104

Goncalves, M., & Tineo, L. (2007). Un Nuevo Paso hacia XQuery Flexible A New Step towards Flexible XQuery. *Revista Avances en Sistemas e Informática, 4*(3), 27-34.

Goncalves, M., & Tineo, L. (2010). Fuzzy XQuery. In Z. Ma, & L. Yan, *Soft Computing in XML Data Management* (pp. 133-163). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-14010-5_6

Goncalves, M., & Tineo, L. (2005). Derivation Principle in Advanced Fuzzy Queries. *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ '05* (pp. 579-584). Reno, NV, USA: IEEE. doi:10.1109/FUZZY.2005.1452458

Jin , Y., & Veerappan, S. (2010). A fuzzy XML database system: Data storage and query processing. *2010 IEEE International Conference on Information Reuse & Integration* (pp. 318-321). Las Vegas, NV, USA: IEEE. doi:10.1109/IRI.2010.5558919

Kansomkeat, S., Sukpisit, S., Thadadech, A., Sae Ueng, P., & Skrbic, S. (2015). Fuzzy ordering implementation applied in fuzzy XQuery. *the 5th International Conference on Information Society and Technology (ICIST 2015),* (pp. 443-493). Kopaonik, Serbia.

Labbad, J. Á., Monascal, R. R., & Tineo, L. (2016). Fuzzy XQuery: A Real Implementation. In L. Yan, *Handbook of Research on Innovative Database Query Processing Techniques* (pp. 158-198). doi:10.4018/978-1-4666-8767-7

Li, T., & Ma, Z. (2018). A structure-based approach of keyword querying for fuzzy XML data. *International Journal of Knowledge-based and Intelligent Engineering Systems, 22*(2), 125-140. doi:10.3233/KES-180379

Lo, A., Kianmehr, K., Kaya, M., Ozyer, T., & Alhajj, R. (2007). Wrapping VRXQuery with Self-Adaptive Fuzzy Capabilities. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)* (pp. 750-756). Fremont, CA, USA: IEEE. doi:10.1109/WI.2007.127

Ma, Z. M., Li, J., & Yan, L. (2010). Fuzzy data modeling and algebraic operations in XML. *INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, 25*(9), 925-947. doi:10.1002/int.20424

Oliboni, B., & Pozzani, G. (2008). Representing Fuzzy Information by Using XML Schema. *2008 19th International Workshop on Database and Expert Systems Applications* (pp. 683-687). Turin, Italy: IEEE. doi:10.1109/DEXA.2008.44

Pak, I. (2005). The area of cyclic polygons: Recent progress on Robbins' conjectures. *Advances in Applied Mathematics, 34*(4), 690-696. Retrieved from https://doi.org/10.1016/j.aam.2004.08.006

Panić, G., Racković, M., & Škrbić, S. (2014). Fuzzy XML and prioritized fuzzy XQuery with implementation. *Journal of Intelligent & Fuzzy Systems, 26*(1), 303-316. doi:10.3233/IFS-120739

Parr, T. (2007). *The Definitive ANTLR Reference: Building Domain-Specific Languages.* North Carolina Dallas, Texas, United States of America: The Pragmatic Bookshelf.

Parr, T. (2010). *ANTLR v3*. Retrieved September 14, 2020, from https://www.antlr3.org/

Sae Ueng, P., Škrbić, S., Kansomkeat, S., & Apirada, T. (2017). A GPFCSP-based fuzzy XQuery interpreter. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* 9(2-2), 35-40.

Sae-Ueng, P., & Skrbic, S. (2020). Priority fuzzy database management system implementation based on extensions to the XQuery language. *Journal of Intelligent & Fuzzy Systems*, 38(4), 4107-4118. doi:10.3233/JIFS-190202

Škrbić, S., & Racković, M. (2013). *Fuzzy Database.* Novi Sad: Faculty of Science, University of Novi Sad.

Škrbić, S., Racković, M., & Takači, A. (2011). Towards the methodology for development of fuzzy relational database applications. *Computer Science and Information Systems, 8*(1), 27-40. doi:10.2298/CSIS100102010S

Škrbić, S., Racković, M., & Takači, A. (2013). Prioritized fuzzy logic based information processing in relational databases. *Knowledge-Based Systems, 38*, 62-73. doi:https://doi.org/10.1016/j.knosys.2012.01.017

Sommerville, I. (2016). *Software Engineering.* Pearson Education Limited.

Spring Boot. (2020). Retrieved September 27, 2020, from Spring: https://spring.io/projects/spring-boot

Sukpisit, S. (2015). *Automated Fuzzy Set Operations for XML Database*. Master's Thesis, Prince of Songkla University, Songkhla, Thailand, p.34. Retrieved from https://kb.psu.ac.th/psukb/bitstream/2016/10701/1/404601.pdf

Sukpisit, S., Kansomkeat, S., Sae Ueng, P., Thadadech, A., & Škrbić, S. (2016). Polygon intersection based algorithm for fuzzy set compatibility calculations. *International Journal of Machine Learning and Computing, 6*, 32-35.

Takači, A. (2005). Schur-concave triangular norms: Characterization and application in pFCSP. *Fuzzy Sets and Systems, 155*(1), 50-64. Retrieved from https://doi.org/10.1016/j.fss.2005.05.011

Takaci, A., Skrbic, S., & Perovic, A. (2009). Generalised Prioritised Fuzzy Constraint Satisfaction Problem. *2009 7th International Symposium on Intelligent Systems and Informatics* (pp. 145-148). Subotica, Serbia: IEEE. doi:10.1109/SISY.2009.5291177

Thadadech, A., Vonghirandecha, P., Kansomkeat, S., & Skrbic, S. (2014). *A Fuzzy XML Database System.* Retrieved from http://kb.psu.ac.th/psukb/bitstream/2016/11260/1/413290.pdf

Üstünkaya, E., Yazici, A., & George, R. (2007). FUZZY DATA REPRESENTATION AND QUERYING IN XML DATABASE. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 15*, 43-57. doi:10.1142/S0218488507004455

Walls, C. (2015). *Spring Boot in Action.* Manning Publications.

Yan, L., Ma, Z., & Zhang, F. (2014). *Fuzzy XML data management.* Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-44899-7

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

Appendix A: EBNF of fuzzy XQuery 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 (',' '$' varname
typedeclaration? ':=' exprsingle)*
	;
whereclause
	:	'where' exprsingle (thresholdexpr)? -> ^('where' exprsingle)
	;
orderbyclause
	:	('order' 'by' | 'stable' 'order' 'by') orderspeclist
	;
orderspeclist
	:	orderspec (',' orderspec)*
	;
orderspec
	:	exprsingle ordermodifier
	;
ordermodifier
	:	('ascending' | 'descending')? ('empty' 'greatest'| 'empty' 'least')?
('collation' uriliteral)?
	;
returnclause
	:	'return'^ '$' 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 // add 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
      ;
integerliteral
      :DIGITS
```

```
	;
decimalliteral
	:(DIGITS '.' DIGITS)
	;
degreeliteral
	: '0.' DIGITS
	;
predefinedentityref
	: '&' ('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: The example of Test cases

Table B.1 Test cases of T2 test scenario with Conjunction "AND"

(A) No priority expression

(B) Both conditions have priority expression

(C) Priority expression is in the first condition

(D) Priority expression is in the second condition

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_01	Triangle	AND	Trapezoidal	(A)	=	=	Crisp number	Crisp number
T2_02	Triangle	AND	Trapezoidal	(A)	=	=	Triangle	Triangle
T2_03	Triangle	AND	Trapezoidal	(A)	=	=	Trapezoidal	Trapezoidal
T2_04	Triangle	AND	Trapezoidal	(A)	=	=	Interval	Interval
T2_05	Triangle	AND	Trapezoidal	(A)	=	=	Leftshoulder	Leftshoulder
T2_06	Triangle	AND	Trapezoidal	(A)	=	=	Rightshoulder	Rightshoulder
T2_07	Triangle	AND	Trapezoidal	(A)	=	!=	Crisp number	Crisp number
T2_08	Triangle	AND	Trapezoidal	(A)	=	!=	Triangle	Triangle
T2_09	Triangle	AND	Trapezoidal	(A)	=	!=	Trapezoidal	Trapezoidal
T2_10	Triangle	AND	Trapezoidal	(A)	=	!=	Interval	Interval
T2_11	Triangle	AND	Trapezoidal	(A)	=	!=	Leftshoulder	Leftshoulder
T2_12	Triangle	AND	Trapezoidal	(A)	=	!=	Rightshoulder	Rightshoulder
T2_13	Triangle	AND	Trapezoidal	(A)	=	<	Crisp number	Crisp number
T2_14	Triangle	AND	Trapezoidal	(A)	=	<	Triangle	Triangle

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_15	Triangle	AND	Trapezoidal	(A)	=	<	Trapezoidal	Trapezoidal
T2_16	Triangle	AND	Trapezoidal	(A)	=	<	Interval	Interval
T2_17	Triangle	AND	Trapezoidal	(A)	=	<	Leftshoulder	Leftshoulder
T2_18	Triangle	AND	Trapezoidal	(A)	=	<	Rightshoulder	Rightshoulder
T2_19	Triangle	AND	Trapezoidal	(A)	=	>	Crisp number	Crisp number
T2_20	Triangle	AND	Trapezoidal	(A)	=	>	Triangle	Triangle
T2_21	Triangle	AND	Trapezoidal	(A)	=	>	Trapezoidal	Trapezoidal
T2_22	Triangle	AND	Trapezoidal	(A)	=	>	Interval	Interval
T2_23	Triangle	AND	Trapezoidal	(A)	=	>	Leftshoulder	Leftshoulder
T2_24	Triangle	AND	Trapezoidal	(A)	=	>	Rightshoulder	Rightshoulder
T2_25	Triangle	AND	Trapezoidal	(A)	!=	=	Crisp number	Crisp number
T2_26	Triangle	AND	Trapezoidal	(A)	!=	=	Triangle	Triangle
T2_27	Triangle	AND	Trapezoidal	(A)	!=	=	Trapezoidal	Trapezoidal
T2_28	Triangle	AND	Trapezoidal	(A)	!=	=	Interval	Interval
T2_29	Triangle	AND	Trapezoidal	(A)	!=	=	Leftshoulder	Leftshoulder
T2_30	Triangle	AND	Trapezoidal	(A)	!=	=	Rightshoulder	Rightshoulder
T2_31	Triangle	AND	Trapezoidal	(A)	!=	!=	Crisp number	Crisp number
T2_32	Triangle	AND	Trapezoidal	(A)	!=	!=	Triangle	Triangle
T2_33	Triangle	AND	Trapezoidal	(A)	!=	!=	Trapezoidal	Trapezoidal
T2_34	Triangle	AND	Trapezoidal	(A)	!=	!=	Interval	Interval
T2_35	Triangle	AND	Trapezoidal	(A)	!=	!=	Leftshoulder	Leftshoulder
T2_36	Triangle	AND	Trapezoidal	(A)	!=	!=	Rightshoulder	Rightshoulder
T2_37	Triangle	AND	Trapezoidal	(A)	!=	<	Crisp number	Crisp number
T2_38	Triangle	AND	Trapezoidal	(A)	!=	<	Triangle	Triangle
T2_39	Triangle	AND	Trapezoidal	(A)	!=	<	Trapezoidal	Trapezoidal
T2_40	Triangle	AND	Trapezoidal	(A)	!=	<	Interval	Interval
T2_41	Triangle	AND	Trapezoidal	(A)	!=	<	Leftshoulder	Leftshoulder
T2_42	Triangle	AND	Trapezoidal	(A)	!=	<	Rightshoulder	Rightshoulder

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_43	Triangle	AND	Trapezoidal	(A)	!=	>	Crisp number	Crisp number
T2_44	Triangle	AND	Trapezoidal	(A)	!=	>	Triangle	Triangle
T2_45	Triangle	AND	Trapezoidal	(A)	!=	>	Trapezoidal	Trapezoidal
T2_46	Triangle	AND	Trapezoidal	(A)	!=	>	Interval	Interval
T2_47	Triangle	AND	Trapezoidal	(A)	!=	>	Leftshoulder	Leftshoulder
T2_48	Triangle	AND	Trapezoidal	(A)	!=	>	Rightshoulder	Rightshoulder
T2_49	Triangle	AND	Trapezoidal	(A)	<	=	Crisp number	Crisp number
T2_50	Triangle	AND	Trapezoidal	(A)	<	=	Triangle	Triangle
T2_51	Triangle	AND	Trapezoidal	(A)	<	=	Trapezoidal	Trapezoidal
T2_52	Triangle	AND	Trapezoidal	(A)	<	=	Interval	Interval
T2_53	Triangle	AND	Trapezoidal	(A)	<	=	Leftshoulder	Leftshoulder
T2_54	Triangle	AND	Trapezoidal	(A)	<	=	Rightshoulder	Rightshoulder
T2_55	Triangle	AND	Trapezoidal	(A)	<	!=	Crisp number	Crisp number
T2_56	Triangle	AND	Trapezoidal	(A)	<	!=	Triangle	Triangle
T2_57	Triangle	AND	Trapezoidal	(A)	<	!=	Trapezoidal	Trapezoidal
T2_58	Triangle	AND	Trapezoidal	(A)	<	!=	Interval	Interval
T2_59	Triangle	AND	Trapezoidal	(A)	<	!=	Leftshoulder	Leftshoulder
T2_60	Triangle	AND	Trapezoidal	(A)	<	!=	Rightshoulder	Rightshoulder
T2_61	Triangle	AND	Trapezoidal	(A)	<	<	Crisp number	Crisp number
T2_62	Triangle	AND	Trapezoidal	(A)	<	<	Triangle	Triangle
T2_63	Triangle	AND	Trapezoidal	(A)	<	<	Trapezoidal	Trapezoidal
T2_64	Triangle	AND	Trapezoidal	(A)	<	<	Interval	Interval
T2_65	Triangle	AND	Trapezoidal	(A)	<	<	Leftshoulder	Leftshoulder
T2_66	Triangle	AND	Trapezoidal	(A)	<	<	Rightshoulder	Rightshoulder
T2_67	Triangle	AND	Trapezoidal	(A)	<	>	Crisp number	Crisp number
T2_68	Triangle	AND	Trapezoidal	(A)	<	>	Triangle	Triangle
T2_69	Triangle	AND	Trapezoidal	(A)	<	>	Trapezoidal	Trapezoidal
T2_70	Triangle	AND	Trapezoidal	(A)	<	>	Interval	Interval

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_71	Triangle	AND	Trapezoidal	(A)	<	>	Leftshoulder	Leftshoulder
T2_72	Triangle	AND	Trapezoidal	(A)	<	>	Rightshoulder	Rightshoulder
T2_73	Triangle	AND	Trapezoidal	(A)	>	=	Crisp number	Crisp number
T2_74	Triangle	AND	Trapezoidal	(A)	>	=	Triangle	Triangle
T2_75	Triangle	AND	Trapezoidal	(A)	>	=	Trapezoidal	Trapezoidal
T2_76	Triangle	AND	Trapezoidal	(A)	>	=	Interval	Interval
T2_77	Triangle	AND	Trapezoidal	(A)	>	=	Leftshoulder	Leftshoulder
T2_78	Triangle	AND	Trapezoidal	(A)	>	=	Rightshoulder	Rightshoulder
T2_79	Triangle	AND	Trapezoidal	(A)	>	!=	Crisp number	Crisp number
T2_80	Triangle	AND	Trapezoidal	(A)	>	!=	Triangle	Triangle
T2_81	Triangle	AND	Trapezoidal	(A)	>	!=	Trapezoidal	Trapezoidal
T2_82	Triangle	AND	Trapezoidal	(A)	>	!=	Interval	Interval
T2_83	Triangle	AND	Trapezoidal	(A)	>	!=	Leftshoulder	Leftshoulder
T2_84	Triangle	AND	Trapezoidal	(A)	>	!=	Rightshoulder	Rightshoulder
T2_85	Triangle	AND	Trapezoidal	(A)	>	<	Crisp number	Crisp number
T2_86	Triangle	AND	Trapezoidal	(A)	>	<	Triangle	Triangle
T2_87	Triangle	AND	Trapezoidal	(A)	>	<	Trapezoidal	Trapezoidal
T2_88	Triangle	AND	Trapezoidal	(A)	>	<	Interval	Interval
T2_89	Triangle	AND	Trapezoidal	(A)	>	<	Leftshoulder	Leftshoulder
T2_90	Triangle	AND	Trapezoidal	(A)	>	<	Rightshoulder	Rightshoulder
T2_91	Triangle	AND	Trapezoidal	(A)	>	>	Crisp number	Crisp number
T2_92	Triangle	AND	Trapezoidal	(A)	>	>	Triangle	Triangle
T2_93	Triangle	AND	Trapezoidal	(A)	>	>	Trapezoidal	Trapezoidal
T2_94	Triangle	AND	Trapezoidal	(A)	>	>	Interval	Interval
T2_95	Triangle	AND	Trapezoidal	(A)	>	>	Leftshoulder	Leftshoulder
T2_96	Triangle	AND	Trapezoidal	(A)	>	>	Rightshoulder	Rightshoulder
T2_97	Triangle	AND	Trapezoidal	(B)	=	=	Crisp number	Crisp number
T2_98	Triangle	AND	Trapezoidal	(B)	=	=	Triangle	Triangle

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_99	Triangle	AND	Trapezoidal	(B)	=	=	Trapezoidal	Trapezoidal
T2_100	Triangle	AND	Trapezoidal	(B)	=	=	Interval	Interval
T2_101	Triangle	AND	Trapezoidal	(B)	=	=	Leftshoulder	Leftshoulder
T2_102	Triangle	AND	Trapezoidal	(B)	=	=	Rightshoulder	Rightshoulder
T2_103	Triangle	AND	Trapezoidal	(B)	=	!=	Crisp number	Crisp number
T2_104	Triangle	AND	Trapezoidal	(B)	=	!=	Triangle	Triangle
T2_105	Triangle	AND	Trapezoidal	(B)	=	!=	Trapezoidal	Trapezoidal
T2_106	Triangle	AND	Trapezoidal	(B)	=	!=	Interval	Interval
T2_107	Triangle	AND	Trapezoidal	(B)	=	!=	Leftshoulder	Leftshoulder
T2_108	Triangle	AND	Trapezoidal	(B)	=	!=	Rightshoulder	Rightshoulder
T2_109	Triangle	AND	Trapezoidal	(B)	=	<	Crisp number	Crisp number
T2_110	Triangle	AND	Trapezoidal	(B)	=	<	Triangle	Triangle
T2_111	Triangle	AND	Trapezoidal	(B)	=	<	Trapezoidal	Trapezoidal
T2_112	Triangle	AND	Trapezoidal	(B)	=	<	Interval	Interval
T2_113	Triangle	AND	Trapezoidal	(B)	=	<	Leftshoulder	Leftshoulder
T2_114	Triangle	AND	Trapezoidal	(B)	=	<	Rightshoulder	Rightshoulder
T2_115	Triangle	AND	Trapezoidal	(B)	=	>	Crisp number	Crisp number
T2_116	Triangle	AND	Trapezoidal	(B)	=	>	Triangle	Triangle
T2_117	Triangle	AND	Trapezoidal	(B)	=	>	Trapezoidal	Trapezoidal
T2_118	Triangle	AND	Trapezoidal	(B)	=	>	Interval	Interval
T2_119	Triangle	AND	Trapezoidal	(B)	=	>	Leftshoulder	Leftshoulder
T2_120	Triangle	AND	Trapezoidal	(B)	=	>	Rightshoulder	Rightshoulder
T2_121	Triangle	AND	Trapezoidal	(B)	!=	=	Crisp number	Crisp number
T2_122	Triangle	AND	Trapezoidal	(B)	!=	=	Triangle	Triangle
T2_123	Triangle	AND	Trapezoidal	(B)	!=	=	Trapezoidal	Trapezoidal
T2_124	Triangle	AND	Trapezoidal	(B)	!=	=	Interval	Interval
T2_125	Triangle	AND	Trapezoidal	(B)	!=	=	Leftshoulder	Leftshoulder
T2_126	Triangle	AND	Trapezoidal	(B)	!=	=	Rightshoulder	Rightshoulder

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_127	Triangle	AND	Trapezoidal	(B)	!=	!=	Crisp number	Crisp number
T2_128	Triangle	AND	Trapezoidal	(B)	!=	!=	Triangle	Triangle
T2_129	Triangle	AND	Trapezoidal	(B)	!=	!=	Trapezoidal	Trapezoidal
T2_130	Triangle	AND	Trapezoidal	(B)	!=	!=	Interval	Interval
T2_131	Triangle	AND	Trapezoidal	(B)	!=	!=	Leftshoulder	Leftshoulder
T2_132	Triangle	AND	Trapezoidal	(B)	!=	!=	Rightshoulder	Rightshoulder
T2_133	Triangle	AND	Trapezoidal	(B)	!=	<	Crisp number	Crisp number
T2_134	Triangle	AND	Trapezoidal	(B)	!=	<	Triangle	Triangle
T2_135	Triangle	AND	Trapezoidal	(B)	!=	<	Trapezoidal	Trapezoidal
T2_136	Triangle	AND	Trapezoidal	(B)	!=	<	Interval	Interval
T2_137	Triangle	AND	Trapezoidal	(B)	!=	<	Leftshoulder	Leftshoulder
T2_138	Triangle	AND	Trapezoidal	(B)	!=	<	Rightshoulder	Rightshoulder
T2_139	Triangle	AND	Trapezoidal	(B)	!=	>	Crisp number	Crisp number
T2_140	Triangle	AND	Trapezoidal	(B)	!=	>	Triangle	Triangle
T2_141	Triangle	AND	Trapezoidal	(B)	!=	>	Trapezoidal	Trapezoidal
T2_142	Triangle	AND	Trapezoidal	(B)	!=	>	Interval	Interval
T2_143	Triangle	AND	Trapezoidal	(B)	!=	>	Leftshoulder	Leftshoulder
T2_144	Triangle	AND	Trapezoidal	(B)	!=	>	Rightshoulder	Rightshoulder
T2_145	Triangle	AND	Trapezoidal	(B)	<	=	Crisp number	Crisp number
T2_146	Triangle	AND	Trapezoidal	(B)	<	=	Triangle	Triangle
T2_147	Triangle	AND	Trapezoidal	(B)	<	=	Trapezoidal	Trapezoidal
T2_148	Triangle	AND	Trapezoidal	(B)	<	=	Interval	Interval
T2_149	Triangle	AND	Trapezoidal	(B)	<	=	Leftshoulder	Leftshoulder
T2_150	Triangle	AND	Trapezoidal	(B)	<	=	Rightshoulder	Rightshoulder
T2_151	Triangle	AND	Trapezoidal	(B)	<	!=	Crisp number	Crisp number
T2_152	Triangle	AND	Trapezoidal	(B)	<	!=	Triangle	Triangle
T2_153	Triangle	AND	Trapezoidal	(B)	<	!=	Trapezoidal	Trapezoidal
T2_154	Triangle	AND	Trapezoidal	(B)	<	!=	Interval	Interval

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_155	Triangle	AND	Trapezoidal	(B)	<	!=	Leftshoulder	Leftshoulder
T2_156	Triangle	AND	Trapezoidal	(B)	<	!=	Rightshoulder	Rightshoulder
T2_157	Triangle	AND	Trapezoidal	(B)	<	<	Crisp number	Crisp number
T2_158	Triangle	AND	Trapezoidal	(B)	<	<	Triangle	Triangle
T2_159	Triangle	AND	Trapezoidal	(B)	<	<	Trapezoidal	Trapezoidal
T2_160	Triangle	AND	Trapezoidal	(B)	<	<	Interval	Interval
T2_161	Triangle	AND	Trapezoidal	(B)	<	<	Leftshoulder	Leftshoulder
T2_162	Triangle	AND	Trapezoidal	(B)	<	<	Rightshoulder	Rightshoulder
T2_163	Triangle	AND	Trapezoidal	(B)	<	>	Crisp number	Crisp number
T2_164	Triangle	AND	Trapezoidal	(B)	<	>	Triangle	Triangle
T2_165	Triangle	AND	Trapezoidal	(B)	<	>	Trapezoidal	Trapezoidal
T2_166	Triangle	AND	Trapezoidal	(B)	<	>	Interval	Interval
T2_167	Triangle	AND	Trapezoidal	(B)	<	>	Leftshoulder	Leftshoulder
T2_168	Triangle	AND	Trapezoidal	(B)	<	>	Rightshoulder	Rightshoulder
T2_169	Triangle	AND	Trapezoidal	(B)	>	=	Crisp number	Crisp number
T2_170	Triangle	AND	Trapezoidal	(B)	>	=	Triangle	Triangle
T2_171	Triangle	AND	Trapezoidal	(B)	>	=	Trapezoidal	Trapezoidal
T2_172	Triangle	AND	Trapezoidal	(B)	>	=	Interval	Interval
T2_173	Triangle	AND	Trapezoidal	(B)	>	=	Leftshoulder	Leftshoulder
T2_174	Triangle	AND	Trapezoidal	(B)	>	=	Rightshoulder	Rightshoulder
T2_175	Triangle	AND	Trapezoidal	(B)	>	!=	Crisp number	Crisp number
T2_176	Triangle	AND	Trapezoidal	(B)	>	!=	Triangle	Triangle
T2_177	Triangle	AND	Trapezoidal	(B)	>	!=	Trapezoidal	Trapezoidal
T2_178	Triangle	AND	Trapezoidal	(B)	>	!=	Interval	Interval
T2_179	Triangle	AND	Trapezoidal	(B)	>	!=	Leftshoulder	Leftshoulder
T2_180	Triangle	AND	Trapezoidal	(B)	>	!=	Rightshoulder	Rightshoulder
T2_181	Triangle	AND	Trapezoidal	(B)	>	<	Crisp number	Crisp number
T2_182	Triangle	AND	Trapezoidal	(B)	>	<	Triangle	Triangle

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_183	Triangle	AND	Trapezoidal	(B)	>	<	Trapezoidal	Trapezoidal
T2_184	Triangle	AND	Trapezoidal	(B)	>	<	Interval	Interval
T2_185	Triangle	AND	Trapezoidal	(B)	>	<	Leftshoulder	Leftshoulder
T2_186	Triangle	AND	Trapezoidal	(B)	>	<	Rightshoulder	Rightshoulder
T2_187	Triangle	AND	Trapezoidal	(B)	>	>	Crisp number	Crisp number
T2_188	Triangle	AND	Trapezoidal	(B)	>	>	Triangle	Triangle
T2_189	Triangle	AND	Trapezoidal	(B)	>	>	Trapezoidal	Trapezoidal
T2_190	Triangle	AND	Trapezoidal	(B)	>	>	Interval	Interval
T2_191	Triangle	AND	Trapezoidal	(B)	>	>	Leftshoulder	Leftshoulder
T2_192	Triangle	AND	Trapezoidal	(B)	>	>	Rightshoulder	Rightshoulder
T2_193	Triangle	AND	Trapezoidal	(C)	=	=	Crisp number	Crisp number
T2_194	Triangle	AND	Trapezoidal	(C)	=	=	Triangle	Triangle
T2_195	Triangle	AND	Trapezoidal	(C)	=	=	Trapezoidal	Trapezoidal
T2_196	Triangle	AND	Trapezoidal	(C)	=	=	Interval	Interval
T2_197	Triangle	AND	Trapezoidal	(C)	=	=	Leftshoulder	Leftshoulder
T2_198	Triangle	AND	Trapezoidal	(C)	=	=	Rightshoulder	Rightshoulder
T2_199	Triangle	AND	Trapezoidal	(C)	=	!=	Crisp number	Crisp number
T2_200	Triangle	AND	Trapezoidal	(C)	=	!=	Triangle	Triangle
T2_201	Triangle	AND	Trapezoidal	(C)	=	!=	Trapezoidal	Trapezoidal
T2_202	Triangle	AND	Trapezoidal	(C)	=	!=	Interval	Interval
T2_203	Triangle	AND	Trapezoidal	(C)	=	!=	Leftshoulder	Leftshoulder
T2_204	Triangle	AND	Trapezoidal	(C)	=	!=	Rightshoulder	Rightshoulder
T2_205	Triangle	AND	Trapezoidal	(C)	=	<	Crisp number	Crisp number
T2_206	Triangle	AND	Trapezoidal	(C)	=	<	Triangle	Triangle
T2_207	Triangle	AND	Trapezoidal	(C)	=	<	Trapezoidal	Trapezoidal
T2_208	Triangle	AND	Trapezoidal	(C)	=	<	Interval	Interval
T2_209	Triangle	AND	Trapezoidal	(C)	=	<	Leftshoulder	Leftshoulder
T2_210	Triangle	AND	Trapezoidal	(C)	=	<	Rightshoulder	Rightshoulder

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_211	Triangle	AND	Trapezoidal	(C)	=	>	Crisp number	Crisp number
T2_212	Triangle	AND	Trapezoidal	(C)	=	>	Triangle	Triangle
T2_213	Triangle	AND	Trapezoidal	(C)	=	>	Trapezoidal	Trapezoidal
T2_214	Triangle	AND	Trapezoidal	(C)	=	>	Interval	Interval
T2_215	Triangle	AND	Trapezoidal	(C)	=	>	Leftshoulder	Leftshoulder
T2_216	Triangle	AND	Trapezoidal	(C)	=	>	Rightshoulder	Rightshoulder
T2_217	Triangle	AND	Trapezoidal	(C)	!=	=	Crisp number	Crisp number
T2_218	Triangle	AND	Trapezoidal	(C)	!=	=	Triangle	Triangle
T2_219	Triangle	AND	Trapezoidal	(C)	!=	=	Trapezoidal	Trapezoidal
T2_220	Triangle	AND	Trapezoidal	(C)	!=	=	Interval	Interval
T2_221	Triangle	AND	Trapezoidal	(C)	!=	=	Leftshoulder	Leftshoulder
T2_222	Triangle	AND	Trapezoidal	(C)	!=	=	Rightshoulder	Rightshoulder
T2_223	Triangle	AND	Trapezoidal	(C)	!=	!=	Crisp number	Crisp number
T2_224	Triangle	AND	Trapezoidal	(C)	!=	!=	Triangle	Triangle
T2_225	Triangle	AND	Trapezoidal	(C)	!=	!=	Trapezoidal	Trapezoidal
T2_226	Triangle	AND	Trapezoidal	(C)	!=	!=	Interval	Interval
T2_227	Triangle	AND	Trapezoidal	(C)	!=	!=	Leftshoulder	Leftshoulder
T2_228	Triangle	AND	Trapezoidal	(C)	!=	!=	Rightshoulder	Rightshoulder
T2_229	Triangle	AND	Trapezoidal	(C)	!=	<	Crisp number	Crisp number
T2_230	Triangle	AND	Trapezoidal	(C)	!=	<	Triangle	Triangle
T2_231	Triangle	AND	Trapezoidal	(C)	!=	<	Trapezoidal	Trapezoidal
T2_232	Triangle	AND	Trapezoidal	(C)	!=	<	Interval	Interval
T2_233	Triangle	AND	Trapezoidal	(C)	!=	<	Leftshoulder	Leftshoulder
T2_234	Triangle	AND	Trapezoidal	(C)	!=	<	Rightshoulder	Rightshoulder
T2_235	Triangle	AND	Trapezoidal	(C)	!=	>	Crisp number	Crisp number
T2_236	Triangle	AND	Trapezoidal	(C)	!=	>	Triangle	Triangle
T2_237	Triangle	AND	Trapezoidal	(C)	!=	>	Trapezoidal	Trapezoidal
T2_238	Triangle	AND	Trapezoidal	(C)	!=	>	Interval	Interval

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_239	Triangle	AND	Trapezoidal	(C)	!=	>	Leftshoulder	Leftshoulder
T2_240	Triangle	AND	Trapezoidal	(C)	!=	>	Rightshoulder	Rightshoulder
T2_241	Triangle	AND	Trapezoidal	(C)	<	=	Crisp number	Crisp number
T2_242	Triangle	AND	Trapezoidal	(C)	<	=	Triangle	Triangle
T2_243	Triangle	AND	Trapezoidal	(C)	<	=	Trapezoidal	Trapezoidal
T2_244	Triangle	AND	Trapezoidal	(C)	<	=	Interval	Interval
T2_245	Triangle	AND	Trapezoidal	(C)	<	=	Leftshoulder	Leftshoulder
T2_246	Triangle	AND	Trapezoidal	(C)	<	=	Rightshoulder	Rightshoulder
T2_247	Triangle	AND	Trapezoidal	(C)	<	!=	Crisp number	Crisp number
T2_248	Triangle	AND	Trapezoidal	(C)	<	!=	Triangle	Triangle
T2_249	Triangle	AND	Trapezoidal	(C)	<	!=	Trapezoidal	Trapezoidal
T2_250	Triangle	AND	Trapezoidal	(C)	<	!=	Interval	Interval
T2_251	Triangle	AND	Trapezoidal	(C)	<	!=	Leftshoulder	Leftshoulder
T2_252	Triangle	AND	Trapezoidal	(C)	<	!=	Rightshoulder	Rightshoulder
T2_253	Triangle	AND	Trapezoidal	(C)	<	<	Crisp number	Crisp number
T2_254	Triangle	AND	Trapezoidal	(C)	<	<	Triangle	Triangle
T2_255	Triangle	AND	Trapezoidal	(C)	<	<	Trapezoidal	Trapezoidal
T2_256	Triangle	AND	Trapezoidal	(C)	<	<	Interval	Interval
T2_257	Triangle	AND	Trapezoidal	(C)	<	<	Leftshoulder	Leftshoulder
T2_258	Triangle	AND	Trapezoidal	(C)	<	<	Rightshoulder	Rightshoulder
T2_259	Triangle	AND	Trapezoidal	(C)	<	>	Crisp number	Crisp number
T2_260	Triangle	AND	Trapezoidal	(C)	<	>	Triangle	Triangle
T2_261	Triangle	AND	Trapezoidal	(C)	<	>	Trapezoidal	Trapezoidal
T2_262	Triangle	AND	Trapezoidal	(C)	<	>	Interval	Interval
T2_263	Triangle	AND	Trapezoidal	(C)	<	>	Leftshoulder	Leftshoulder
T2_264	Triangle	AND	Trapezoidal	(C)	<	>	Rightshoulder	Rightshoulder
T2_265	Triangle	AND	Trapezoidal	(C)	>	=	Crisp number	Crisp number
T2_266	Triangle	AND	Trapezoidal	(C)	>	=	Triangle	Triangle

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_267	Triangle	AND	Trapezoidal	(C)	>	=	Trapezoidal	Trapezoidal
T2_268	Triangle	AND	Trapezoidal	(C)	>	=	Interval	Interval
T2_269	Triangle	AND	Trapezoidal	(C)	>	=	Leftshoulder	Leftshoulder
T2_270	Triangle	AND	Trapezoidal	(C)	>	=	Rightshoulder	Rightshoulder
T2_271	Triangle	AND	Trapezoidal	(C)	>	!=	Crisp number	Crisp number
T2_272	Triangle	AND	Trapezoidal	(C)	>	!=	Triangle	Triangle
T2_273	Triangle	AND	Trapezoidal	(C)	>	!=	Trapezoidal	Trapezoidal
T2_274	Triangle	AND	Trapezoidal	(C)	>	!=	Interval	Interval
T2_275	Triangle	AND	Trapezoidal	(C)	>	!=	Leftshoulder	Leftshoulder
T2_276	Triangle	AND	Trapezoidal	(C)	>	!=	Rightshoulder	Rightshoulder
T2_277	Triangle	AND	Trapezoidal	(C)	>	<	Crisp number	Crisp number
T2_278	Triangle	AND	Trapezoidal	(C)	>	<	Triangle	Triangle
T2_278	Triangle	AND	Trapezoidal	(C)	>	<	Trapezoidal	Trapezoidal
T2_280	Triangle	AND	Trapezoidal	(C)	>	<	Interval	Interval
T2_281	Triangle	AND	Trapezoidal	(C)	>	<	Leftshoulder	Leftshoulder
T2_282	Triangle	AND	Trapezoidal	(C)	>	<	Rightshoulder	Rightshoulder
T2_283	Triangle	AND	Trapezoidal	(C)	>	>	Crisp number	Crisp number
T2_284	Triangle	AND	Trapezoidal	(C)	>	>	Triangle	Triangle
T2_285	Triangle	AND	Trapezoidal	(C)	>	>	Trapezoidal	Trapezoidal
T2_286	Triangle	AND	Trapezoidal	(C)	>	>	Interval	Interval
T2_287	Triangle	AND	Trapezoidal	(C)	>	>	Leftshoulder	Leftshoulder
T2_288	Triangle	AND	Trapezoidal	(C)	>	>	Rightshoulder	Rightshoulder
T2_289	Triangle	AND	Trapezoidal	(D)	=	=	Crisp number	Crisp number
T2_290	Triangle	AND	Trapezoidal	(D)	=	=	Triangle	Triangle
T2_291	Triangle	AND	Trapezoidal	(D)	=	=	Trapezoidal	Trapezoidal
T2_292	Triangle	AND	Trapezoidal	(D)	=	=	Interval	Interval
T2_293	Triangle	AND	Trapezoidal	(D)	=	=	Leftshoulder	Leftshoulder
T2_294	Triangle	AND	Trapezoidal	(D)	=	=	Rightshoulder	Rightshoulder

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_295	Triangle	AND	Trapezoidal	(D)	=	!=	Crisp number	Crisp number
T2_296	Triangle	AND	Trapezoidal	(D)	=	!=	Triangle	Triangle
T2_297	Triangle	AND	Trapezoidal	(D)	=	!=	Trapezoidal	Trapezoidal
T2_298	Triangle	AND	Trapezoidal	(D)	=	!=	Interval	Interval
T2_299	Triangle	AND	Trapezoidal	(D)	=	!=	Leftshoulder	Leftshoulder
T2_300	Triangle	AND	Trapezoidal	(D)	=	!=	Rightshoulder	Rightshoulder
T2_301	Triangle	AND	Trapezoidal	(D)	=	<	Crisp number	Crisp number
T2_302	Triangle	AND	Trapezoidal	(D)	=	<	Triangle	Triangle
T2_303	Triangle	AND	Trapezoidal	(D)	=	<	Trapezoidal	Trapezoidal
T2_304	Triangle	AND	Trapezoidal	(D)	=	<	Interval	Interval
T2_305	Triangle	AND	Trapezoidal	(D)	=	<	Leftshoulder	Leftshoulder
T2_306	Triangle	AND	Trapezoidal	(D)	=	<	Rightshoulder	Rightshoulder
T2_307	Triangle	AND	Trapezoidal	(D)	=	>	Crisp number	Crisp number
T2_308	Triangle	AND	Trapezoidal	(D)	=	>	Triangle	Triangle
T2_309	Triangle	AND	Trapezoidal	(D)	=	>	Trapezoidal	Trapezoidal
T2_310	Triangle	AND	Trapezoidal	(D)	=	>	Interval	Interval
T2_311	Triangle	AND	Trapezoidal	(D)	=	>	Leftshoulder	Leftshoulder
T2_312	Triangle	AND	Trapezoidal	(D)	=	>	Rightshoulder	Rightshoulder
T2_313	Triangle	AND	Trapezoidal	(D)	!=	=	Crisp number	Crisp number
T2_314	Triangle	AND	Trapezoidal	(D)	!=	=	Triangle	Triangle
T2_315	Triangle	AND	Trapezoidal	(D)	!=	=	Trapezoidal	Trapezoidal
T2_316	Triangle	AND	Trapezoidal	(D)	!=	=	Interval	Interval
T2_317	Triangle	AND	Trapezoidal	(D)	!=	=	Leftshoulder	Leftshoulder
T2_318	Triangle	AND	Trapezoidal	(D)	!=	=	Rightshoulder	Rightshoulder
T2_319	Triangle	AND	Trapezoidal	(D)	!=	!=	Crisp number	Crisp number
T2_320	Triangle	AND	Trapezoidal	(D)	!=	!=	Triangle	Triangle
T2_321	Triangle	AND	Trapezoidal	(D)	!=	!=	Trapezoidal	Trapezoidal
T2_322	Triangle	AND	Trapezoidal	(D)	!=	!=	Interval	Interval

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_323	Triangle	AND	Trapezoidal	(D)	!=	!=	Leftshoulder	Leftshoulder
T2_324	Triangle	AND	Trapezoidal	(D)	!=	!=	Rightshoulder	Rightshoulder
T2_325	Triangle	AND	Trapezoidal	(D)	!=	<	Crisp number	Crisp number
T2_326	Triangle	AND	Trapezoidal	(D)	!=	<	Triangle	Triangle
T2_327	Triangle	AND	Trapezoidal	(D)	!=	<	Trapezoidal	Trapezoidal
T2_328	Triangle	AND	Trapezoidal	(D)	!=	<	Interval	Interval
T2_329	Triangle	AND	Trapezoidal	(D)	!=	<	Leftshoulder	Leftshoulder
T2_330	Triangle	AND	Trapezoidal	(D)	!=	<	Rightshoulder	Rightshoulder
T2_331	Triangle	AND	Trapezoidal	(D)	!=	>	Crisp number	Crisp number
T2_332	Triangle	AND	Trapezoidal	(D)	!=	>	Triangle	Triangle
T2_333	Triangle	AND	Trapezoidal	(D)	!=	>	Trapezoidal	Trapezoidal
T2_334	Triangle	AND	Trapezoidal	(D)	!=	>	Interval	Interval
T2_335	Triangle	AND	Trapezoidal	(D)	!=	>	Leftshoulder	Leftshoulder
T2_336	Triangle	AND	Trapezoidal	(D)	!=	>	Rightshoulder	Rightshoulder
T2_337	Triangle	AND	Trapezoidal	(D)	<	=	Crisp number	Crisp number
T2_338	Triangle	AND	Trapezoidal	(D)	<	=	Triangle	Triangle
T2_339	Triangle	AND	Trapezoidal	(D)	<	=	Trapezoidal	Trapezoidal
T2_340	Triangle	AND	Trapezoidal	(D)	<	=	Interval	Interval
T2_341	Triangle	AND	Trapezoidal	(D)	<	=	Leftshoulder	Leftshoulder
T2_342	Triangle	AND	Trapezoidal	(D)	<	=	Rightshoulder	Rightshoulder
T2_343	Triangle	AND	Trapezoidal	(D)	<	!=	Crisp number	Crisp number
T2_344	Triangle	AND	Trapezoidal	(D)	<	!=	Triangle	Triangle
T2_345	Triangle	AND	Trapezoidal	(D)	<	!=	Trapezoidal	Trapezoidal
T2_346	Triangle	AND	Trapezoidal	(D)	<	!=	Interval	Interval
T2_347	Triangle	AND	Trapezoidal	(D)	<	!=	Leftshoulder	Leftshoulder
T2_348	Triangle	AND	Trapezoidal	(D)	<	!=	Rightshoulder	Rightshoulder
T2_349	Triangle	AND	Trapezoidal	(D)	<	<	Crisp number	Crisp number
T2_350	Triangle	AND	Trapezoidal	(D)	<	<	Triangle	Triangle

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_351	Triangle	AND	Trapezoidal	(D)	<	<	Trapezoidal	Trapezoidal
T2_352	Triangle	AND	Trapezoidal	(D)	<	<	Interval	Interval
T2_353	Triangle	AND	Trapezoidal	(D)	<	<	Leftshoulder	Leftshoulder
T2_354	Triangle	AND	Trapezoidal	(D)	<	<	Rightshoulder	Rightshoulder
T2_355	Triangle	AND	Trapezoidal	(D)	<	>	Crisp number	Crisp number
T2_356	Triangle	AND	Trapezoidal	(D)	<	>	Triangle	Triangle
T2_357	Triangle	AND	Trapezoidal	(D)	<	>	Trapezoidal	Trapezoidal
T2_358	Triangle	AND	Trapezoidal	(D)	<	>	Interval	Interval
T2_359	Triangle	AND	Trapezoidal	(D)	<	>	Leftshoulder	Leftshoulder
T2_360	Triangle	AND	Trapezoidal	(D)	<	>	Rightshoulder	Rightshoulder
T2_361	Triangle	AND	Trapezoidal	(D)	>	=	Crisp number	Crisp number
T2_362	Triangle	AND	Trapezoidal	(D)	>	=	Triangle	Triangle
T2_363	Triangle	AND	Trapezoidal	(D)	>	=	Trapezoidal	Trapezoidal
T2_364	Triangle	AND	Trapezoidal	(D)	>	=	Interval	Interval
T2_365	Triangle	AND	Trapezoidal	(D)	>	=	Leftshoulder	Leftshoulder
T2_366	Triangle	AND	Trapezoidal	(D)	>	=	Rightshoulder	Rightshoulder
T2_367	Triangle	AND	Trapezoidal	(D)	>	!=	Crisp number	Crisp number
T2_368	Triangle	AND	Trapezoidal	(D)	>	!=	Triangle	Triangle
T2_369	Triangle	AND	Trapezoidal	(D)	>	!=	Trapezoidal	Trapezoidal
T2_370	Triangle	AND	Trapezoidal	(D)	>	!=	Interval	Interval
T2_371	Triangle	AND	Trapezoidal	(D)	>	!=	Leftshoulder	Leftshoulder
T2_372	Triangle	AND	Trapezoidal	(D)	>	!=	Rightshoulder	Rightshoulder
T2_373	Triangle	AND	Trapezoidal	(D)	>	<	Crisp number	Crisp number
T2_374	Triangle	AND	Trapezoidal	(D)	>	<	Triangle	Triangle
T2_375	Triangle	AND	Trapezoidal	(D)	>	<	Trapezoidal	Trapezoidal
T2_376	Triangle	AND	Trapezoidal	(D)	>	<	Interval	Interval
T2_377	Triangle	AND	Trapezoidal	(D)	>	<	Leftshoulder	Leftshoulder
T2_378	Triangle	AND	Trapezoidal	(D)	>	<	Rightshoulder	Rightshoulder

Test case ID	First Fuzzy Expression	Conjunction	Second Fuzzy Expression	Priority Expression	First operator	Second operator	First variable expression	Second variable expression
T2_379	Triangle	AND	Trapezoidal	(D)	>	>	Crisp number	Crisp number
T2_380	Triangle	AND	Trapezoidal	(D)	>	>	Triangle	Triangle
T2_381	Triangle	AND	Trapezoidal	(D)	>	>	Trapezoidal	Trapezoidal
T2_382	Triangle	AND	Trapezoidal	(D)	>	>	Interval	Interval
T2_383	Triangle	AND	Trapezoidal	(D)	>	>	Leftshoulder	Leftshoulder
T2_384	Triangle	AND	Trapezoidal	(D)	>	>	Rightshoulder	Rightshoulder

Table B.2 Example of Test case values

T2. The first condition is *triangle fuzzy number* and the second condition is *trapezoidal fuzzy number*, with operator AND in the query.

I. The fuzzy XQuery does not have any priority expression.

a. The first relational operator is = and the second relational operator is = (= AND =).

Fuzzy XQuery condition: **where $x/height = #tri(170,180,190)# AND $x/age = #trap(30,40,50,60)#**

Test case ID	Test data		Expected Result (Satisfaction degree)	Actual Result (Satisfaction degree)	Status
	$x/height	$x/age			
T2_01	160	45	0	0	Pass
T2_02	Tri(140,150,160)	Tri(30,45,60)	0	0	Pass
T2_03	Trap(160,170,190,200)	Trap(20,25,30,35)	0	0	Pass
T2_04	Interval(170,180)	Interval(65,70)	0	0	Pass
T2_05	Leftshoulder(180,190)	Leftshoulder(25,30)	0	0	Pass
T2_06	Rightshoulder(180,190)	Rightshoulder(25,30)	0	0	Pass

b. The first relational operator is = and the second relational operator is != (= AND !=).

Fuzzy XQuery condition: **where $x/height = #tri(170,180,190)# AND $x/age != #trap(30,40,50,60)#**

Test case ID	Test data		Expected Result (Satisfaction degree)	Actual Result (Satisfaction degree)	Status
	$x/height	$x/age			
T2_07	160	45	0	0	Pass
T2_08	Tri(140,150,160)	Tri(30,45,60)	0	0	Pass
T2_09	Trap(160,170,190,200)	Trap(20,25,30,35)	0.247	0.247	Pass
T2_10	Interval(170,180)	Interval(65,70)	0.5	0.5	Pass
T2_11	Leftshoulder(180,190)	Leftshoulder(25,30)	0.054	0.054	Pass
T2_12	Rightshoulder(180,190)	Rightshoulder(25,30)	0	0	Pass

c. The first relational operator is = and the second relational operator is < (= AND !<).

Fuzzy XQuery condition: **where $x/height = #tri(170,180,190)# AND $x/age < #trap(30,40,50,60)#**

Test case ID	Test data		Expected Result (Satisfaction degree)	Actual Result (Satisfaction degree)	Status
	$x/height	$x/age			
T2_13	160	45	0	0	Pass
T2_14	Tri(140,150,160)	Tri(30,45,60)	0	0	Pass
T2_15	Trap(160,170,190,200)	Trap(20,25,30,35)	0.33	0.33	Pass
T2_16	Interval(170,180)	Interval(65,70)	0	0	Pass

T2_17	Leftshoulder(180,190)	Leftshoulder(25,30)	0.054	0.054	Pass
T2_18	Rightshoulder(180,190)	Rightshoulder(25,30)	0	0	Pass

d. The first relational operator is = and the second relational operator is > (= AND >).

Fuzzy XQuery condition: **where $x/height = #tri(170,180,190)# AND $x/age > #trap(30,40,50,60)#**

Test case ID	Test data		Expected Result (Satisfaction degree)	Actual Result (Satisfaction degree)	Status
	$x/height	$x/age			
T2_19	160	45	0	0	Pass
T2_20	Tri(140,150,160)	Tri(30,45,60)	0	0	Pass
T2_21	Trap(160,170,190,200)	Trap(20,25,30,35)	0	0	Pass
T2_22	Interval(170,180)	Interval(65,70)	0.5	0.5	Pass
T2_23	Leftshoulder(180,190)	Leftshoulder(25,30)	0	0	Pass
T2_24	Rightshoulder(180,190)	Rightshoulder(25,30)	0	0	Pass

e. The first relational operator is != and the second relational operator is = (!= AND =).

Fuzzy XQuery condition: **where $x/height != #tri(170,180,190)# AND $x/age = #trap(30,40,50,60)#**

Test case ID	Test data		Expected Result (Satisfaction degree)	Actual Result (Satisfaction degree)	Status
	$x/height	$x/age			
T2_25	160	45	1	1	Pass
T2_26	Tri(140,150,160)	Tri(30,45,60)	1	1	Pass

T2_27	Trap(160,170,190,200)	Trap(20,25,30,35)	0	0	Pass
T2_28	Interval(170,180)	Interval(65,70)	0	0	Pass
T2_29	Leftshoulder(180,190)	Leftshoulder(25,30)	0	0	Pass
T2_30	Rightshoulder(180,190)	Rightshoulder(25,30)	0	0	Pass

УНИВЕРЗИТЕТ У НОВОМ САДУ **ОБРАЗАЦ – 5а**
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Pannipa Sae-Ueng
Ментор (титула, име, презиме, звање, институција)	др Срђан Шкрбић, редовни професор, Универзитет у Новом Саду, Природно математички факултет
Наслов рада:	Development of XQuery Interpreter Extensions Based on Fuzzy Logic with Priorities
Језик публикације (писмо):	Енглески
Физички опис рада:	Унети број: Страница 120 Поглавља 6 Референци 44 Табела 19 Слика 47 Графикона 0 Прилога 2
Научна област:	Информатика
Ужа научна област (научна дисциплина):	Рачунарске науке
Кључне речи / предметна одредница:	Fuzzy XQuery, XQuery Interpreter, XQuery, XML Database
Резиме на језику рада:	In many real-world applications, information is often imprecise and uncertain. With the popularity of web-based applications, huge amounts of data are available on the web, and XML (eXtensible Markup Language) has become the de facto standard for data exchange over the internet. The XQuery is the language for querying XML data. However, XML and XQuery suffer from incapability of representing and manipulating imprecise and uncertain data. Consequently, this work represents fuzzy data in XML documents and extends XQuery language as providing a more flexible XQuery language by using the fuzzy set theory. In this thesis, an extension of the XQuery query, called Fuzzy XQuery is described. It allows users to define priority, threshold and fuzzy expressions in their queries. Users also can predefine linguistic terms to use them in querying. An algorithm for calculating the global constraint satisfaction degree using the Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) is introduced. Furthermore, Fuzzy XQuery Interpreter (FXI) is implemented allowing execution of fuzzy XQuery queries based on open source technologies and native XML open- source database. Additionally, innovative methods for computing fuzzy set compatibility and introducing order over fuzzy sets have been implemented, which give serious improvements in computational performance compared to previous implementations.

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
5г – Изјава о коришћењу.
Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

Датум прихватања теме од стране надлежног већа:	
Датум одбране:	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: др Милош Рацковић, редовни професор, Универзитет у Новом Саду, Природно-математички факултет Ментор: др Срђан Шкрбић, редовни професор, Универзитет у Новом Саду, Природно-математички факултет Члан: Dr. Wiphada Wettayaprasit, assistant professor, Prince of Songkla University, Faculty of Science Члан: др Александар Такачи, редовни професор, Универзитет у Новом Саду, Технолошки факултет
Напомена:	

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Pannipa Sae-Ueng
Supervisor (title, first name, last name, position, institution)	Dr. Srdjan Skrbic, full profesor, Univesity of Novi Sad, Faculty of Sciences
Thesis title:	Development of XQuery Interpreter Extensions Based on Fuzzy Logic with Priorities
Language of text (script):	English
Physical description:	Number of: Pages 120 Chapters 6 References 44 Tables 19 Figures 47 Graphs 0 Appendices 2
Scientific field:	Informatics
Scientific subfield (scientific discipline):	Computer Science
Subject, Key words:	Fuzzy XQuery, XQuery Interpreter, XQuery, XML Database
Abstract in English language:	In many real-world applications, information is often imprecise and uncertain. With the popularity of web-based applications, huge amounts of data are available on the web, and XML (eXtensible Markup Language) has become the de facto standard for data exchange over the internet. The XQuery is the language for querying XML data. However, XML and XQuery suffer from incapability of representing and manipulating imprecise and uncertain data. Consequently, this work represents fuzzy data in XML documents and extends XQuery language as providing a more flexible XQuery language by using the fuzzy set theory. In this thesis, an extension of the XQuery query, called Fuzzy XQuery is described. It allows users to define priority, threshold and fuzzy expressions in their queries. Users also can predefine linguistic terms to use them in querying. An algorithm for calculating the global constraint satisfaction degree using the Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) is introduced. Furthermore, Fuzzy XQuery Interpreter (FXI) is implemented allowing execution of fuzzy XQuery queries based on open source technologies and native XML open- source database. Additionally, innovative methods for computing fuzzy set compatibility and introducing order over fuzzy sets have been implemented, which give serious improvements in computational performance compared to previous implementations.

[2] The author of doctoral dissertation has signed the following Statements:
5б – Statement on the authority,
5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
5г – Statement on copyright licenses.
The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

Accepted on Scientific Board on:	
Defended:	
Thesis Defend Board: (title, first name, last name, position, institution)	President: Dr. Milos Rackovic, full professor, University of Novi Sad, Faculty of Sciences Supervisor: Dr. Srdjan Skrbic, full professor, University of Novi Sad, Faculty of Sciences Member: Dr. Wiphada Wettayaprasit, assistant professor, Prince of Songkla University, Faculty of Science Member: Dr. Aleksandar Takaci, full professor, University of Novi Sad, Faculty of Technology
Note:	

This Form is an integral part of the doctoral dissertation or the doctoral art project that is being defended at the University of Novi Sad. Include the completed form after the text of the doctoral dissertation or the doctoral art project.

Data treatment plan

Project/research title
Development of XQuery Interpreter Extensions Based on Fuzzy Logic with Priorities
Name of the institution / institutions within which the research is conducted
a) Department of Mathematics and informatics, Faculty of Sciences, University of Novi Sad b) c)
The name of the program within which the research is realized
Ph.D. program in informatics
1. Data description

1.1 Type of study

No data were collected in this study.

Briefly describe the type of study in which the data are collected

_____

_____

_____

1.2 Data types
a) quantitative
b) qualitative

1.3. Data collection method
a) polls, questionnaires, tests
b) clinical assessments, medical records, electronic health records
c) genotypes: specify type _____
d) administrative data: specify type _____
e) tissue samples: specify type _____
f) recordings, photographs: specify type _____
g) text, specify type _____
h) map, specify type _____
i) other: describe _____

1.3 Data format, scales used, amount of data

1.3.1 Software used and file format:
a) Excel file _____

b) SPSS file _____
c) PDF file _____
d) Textual file _____
e) JPG file _____
f) Other file _____

1.3.2. Number of records (for quantitative data)

a) number of variables _____
b) number of measurements (respondents, assessment, recordings, etc.) _____

1.3.3. Repeated measurements
a) yes
b) no

If the answer is yes, please answer the following questions:
a) the time interval between repeated measurements is_____
b) variables that are measured multiple times refer to _____
c) new versions of files that contain repeated measurements are named as _____

Notes: _____

Do formats and software enable data sharing and long-term validity?
a) Yes
b) No
If the answer is no, please explain _____
_____

2. Data collection

2.1 Methodology for data collection/generation

2.1.1. Within which research project was the data collected?
a) experiment, specify type _____
b) correlational research, specify type _____
c) text analysis, specify type _____
d) other, specify what _____

2.1.2 Indicate the types of measuring instruments or data standards specific to a particular scientific discipline (if any).
_____

_____

2.2 Data quality and standards

2.2.1. Missing data treatment
a) Does the matrix contain missing data? Yes No

If the answer is yes, please answer the following questions:
a) What is the number of missing data? _____
b) Is the user of the matrix recommended to replace the missing data? Yes No
c) If the answer is yes, provide suggestions for treatment to replace the missing data

_____

2.2.2. How is data quality controlled? Please describe.

_____

_____

2.2.3. How the data entry into the matrix was controlled?

_____

_____

3. Data treatment and supporting documentation

3.1. Treatment and storage of data

3.1.1. Data will be deposited in _____ repository.
3.1.2. URL address _____
3.1.3. DOI _____

3.1.4. Will the data be in open access?
a) Yes
b) Yes, after an embargo that will last until _____
c) No

If the answer is no, please state the reason _____

3.1.5. The data will not be deposited in the repository, but will be kept locally stored. Justification

_____

_____

3.2 Metadata and data documentation
3.2.1. Which metadata standard will be applied? _____

3.2.1. Specify the metadata on the basis of which the data was deposited in the repository.

_____

_____

If necessary, indicate the methods used to retrieve data, analytical and procedural information, their coding, detailed descriptions of variables, records, etc..

_____

_____

_____

_____

3.3 Data retention strategy and standards
3.3.1. Until when will the data be stored in the repository? _____
3.3.2. Will the data be deposited under a password? Yes No
3.3.3. Will the password be available to a particular circle of researchers? Yes No
3.3.4. The data must be removed from open access after some time:
Yes No
Please justify
_____

_____

4. Data security and protection of confidential information

This section MUST be completed if your information includes personal information relating to survey participants. For other research, data protection and security should also be considered.

4.1 Formal standards for information / data security
Researchers conducting human trials must adhere to the Law on Personal Data Protection (*https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html*) and the relevant institutional code on academic integrity.

4.1.2. Has the research been approved by an ethics committee? Yes No
If the answer is Yes, state the date and name of the ethics committee that approved the research
_____

4.1.2. Does the data include the personal data of the research participants? Yes No
If yes, please indicate how you ensured the confidentiality and security of the information related to the respondents:
a) Data is not in open access
b) Data has been anonimized
c) Other, please specify
_____
_____

5. Data availability

5.1. The data will be
a) publicly accessible
б) available only to a narrow circle of researchers in a particular scientific field
ц) closed

If the data are only available to a narrow circle of researchers, indicate the conditions under which they can use them:
_____

_____

If the data are only available to a narrow circle of researchers, indicate how they can access the data:
_____

_____

5.4. Specify the license under which the collected data will be archived.

_____

6. Roles and responsibilities

6.1. Provide the name and e-mail address of the owner (author) of the data

_____

6.2. Provide the name and email address of the person maintaining the data matrix

_____

6.3. Provide the name and email address of the person providing access to the data to other researchers

_____

Short biography

Pannipa Sae-Ueng was born on 22 January 1975 in Trang Province, Thailand. She started her undergraduate studies in Computer Science at Faculty of Sciences, Prince of Songkla University in 1994 and graduated with Second-Class Honours in 1998. From 1998 to 2002 she worked as a system analyst at Computer Center of Walailak University in Nakhon Si Thammarat Province. In the period 2002 – 2004, she was employed as a system analyst at the Government Savings Bank in Bangkok. Between 2004 and 2007 she studied a master degree at Faculty of Engineering, Chulalongkorn University and defended her master thesis entitled "A Development of DSPACE Programming Interface for Center of Academic Resources, Chulalongkorn University".

Since 2008 she has been employed as a lecturer at the Department of Computer Science, Faculty of Science, Prince of Songkla University in Songkhla Province. She teaches courses to students of computer science on the bachelor level. After that in 2011, she received the scholarship from Prince of Songkla University to study Ph.D. at Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Republic of Serbia. She published (as author or co-author) about 10 scientific papers in the field of fuzzy set and fuzzy logic.

Novi Sad, September 2021 Pannipa Sae-Ueng