SATISFIABILITY REASONING OVER VAGUE ONTOLOGIES USING FUZZY SOFT SET THEORY

BY

RIDWAN SALAHUDEEN

DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF PHYSICAL SCIENCES,
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA.

NOVEMBER, 2016

SATISFIABILITY REASONING OVER VAGUE ONTOLOGIES USING FUZZY SOFT SET THEORY

By

Ridwan SALAHUDEEN
B.Sc. Computer Science (ABU), 2010
P13SCMT8017

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A M.Sc. COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF PHYSICAL SCIENCES,
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA.

NOVEMBER, 2016

DECLARATION

I declares that the work in this dissertation entitled "**Satisfiability Reasoning over Vague Ontologies using Fuzzy Soft Set Theory**" has been carried out by me in the Department of Computer Science. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other Institution.

RIDWAN SALAHUDEEN	_____	_____
Name of Student	Signature	Date

CERTIFICATION

This dissertation entitled **SATISFIABILITY REASONING OVER VAGUE ONTOLOGIES USING FUZZY SOFT SET THEORY** by Ridwan SALAHUDEEN meets the regulations governing the award of the degree of M.Sc. Computer Science of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Dr. A. F. Donfack Kana	_____	_____
Chairman, Supervisory Committee	Signature	Date
Dr. S. E. Abdullahi	_____	_____
Member, Supervisory Committee	Signature	Date
Prof. S. B. Junaidu	_____	_____
Head of Department	Signature	Date
_____	_____	_____
Dean, School of Postgraduate Studies	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

ACKNOWLEDGEMENT

ALHAMDULILLAHI! To ALLAH (SWT) be the Glory, I am deeply humbled for the grace and mercy of Allah (SWT) and I fully acknowledge all His favors in my life.

The contributions of many people were vital tools in the successful completion of this research. I believe without their encouragements, supports and contributions this research wouldn't have seen the light of the day. All thanks are due to Allah for using these people on me.

I wish to express my sincere gratitude to my supervisors Dr. A. F. Donfack Kana and Dr. S. E. Abdullahi for their unquantifiable assistance and guidance throughout the research work. With their rich ideas and deep sense of originality this research was able to attain a profound standard.

I wish to express my utmost gratitude to my dearest family for their unconditional love, patience, understanding and encouragement. This is saying a BIG thank you to my wife, Aishah Muhammad, my daughter, Aishah Ummulkhair Ridwan. To my parents, I say, may Allah reward you with *Jannatul Firdous.* To my siblings, I say, I am privileged to have you all as my strength, *what can I do without a family?* May Allah reward you all abundantly (Amin).

I also want to express my appreciation to all my course mate, dear friends within and outside Nigeria for their pleasant advices and encouragements. May Allah grant you all your wishes.

I am personally indebted to the head of my department in person of Prof. S. B. Junaidu who has constantly act as a father to me, and to all the lecturers of Computer Science, Mathematics and Statistics departments who have groomed me with the relevant knowledge towards the achievement of my MSc degree.

ABSTRACT

Ontology is an explicit representation of conceptualizations, which is widely used in modelling real world domains by defining their shared vocabularies such that they can be understood by both human and machines for the purpose of information sharing. Description Logic (DL) is a knowledge representation language that is widely used in building ontologies as well as providing the foundation in which modern web ontologies languages such as OWL are built upon. Classical ontology definitions contain concepts and relations that describe asserted facts about the real world. Earlier studies on ontologies overlooked the representation of uncertainty in their formalizations. However, for a real world domain to be fully modelled, its uncertain aspects must be reflected and appropriately represented. This work propose a satisfiability reasoning algorithm based on fuzzy soft set theory in order to reason about the uncertain aspect of an ontology of vague domain. The proposed algorithm was evaluated by applying it on some vague ontologies and the result was compared with the tableaux based and the soft set ontology reasoning techniques. The obtained result shows that, the proposed algorithm is satisfiable when fuzzy concepts and assertions are involved in an ontology representation while such fuzzy conceptions are not handled by both tableaux based and soft set ontology procedures.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABox	Assertional Box
AI	Artificial Intelligence
ALC	Attribute Language with Complement
BN	Bayesian Network
CPT	Conditional Probability Table
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DL	Description Logic
FSSO	Fuzzy Soft Set Ontology
MEBN	Multi-Entity Bayesian Networks
NNF	Negation Normal Form
OCML	Objective Categorical Abstract Machine Language
OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity
OWL	Web Ontology Language
RBox	Role Box
TBox	Terminology Box
WWW	World Wide Web
W3C	World Wide Web Consortium

Chapter One

Introduction

1.1. Background to the Study

The use of information from heterogeneous sources is an intelligent task that requires human being who has a background knowledge of the information. However, due to existence of several information sources, human processing speed cannot be relied upon for speedy information processing. In contrast, computers can easily deal with such voluminous information as long as their processing do not require human intelligence. Therefore, for information to be processed efficiently, its processing must be automated. Almost all information can be represented in natural language, this richness of natural language, however, makes it very difficult to process computationally. The traditional computational processing of information involved a pattern matching process, a literal character by character comparison of the words in natural languages representing information. This simplistic computational processing approach is known as syntactic information processing. On the contrary, semantics information representation provides a universal understanding of information (both by human and machine) and leads to automated information processing. This can be achieved by attaching meaning to letters, words, phrases, signs, and symbols. Semantic information processing is seen as a means of resolving the problem of ambiguities in syntactic information processing (Richardson, 1994).

While talking about automated processing for natural language, Miller (1995) stated that, because meaningful sentences are composed of meaningful words, any system that hopes to process natural languages as people do must have information about words and their meanings. This information is traditionally provided through dictionaries, and machine-readable

dictionaries are now widely available. But dictionary entries evolved for the convenience of human readers, not for machines.

According to Kana and Akinkunmi (2014), for the semantic processing of information to be possible, systems must be able to understand the meaning of data they are processing and then, perform the processing semantically. To achieve that, three key issues must be resolved:

a) Information should be represented in such a way that, its semantics is contained within its representation and should be unambiguous.
b) There should be a possibility of deducing the semantic of the data represented by machines possibly with some inference capability.
c) There should be a possibility of two or more system processing related information to interoperate.

Traditionally, data representation and processing is only limited to the syntactic level only, which cannot achieve the semantic goal. It is unanimously agreed upon that ontological representation of knowledge is providing the necessary solution for achieving a successful semantic information representation.

A clearer definition of an ontology was provided by Sowa (2000) as:

> "The study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalog of the types of things that are assumed to exist in a domain of interest **D** from the perspective of a person who uses a language **L** for the purpose of talking about **D**. The types in the ontology represent the predicates, word senses, or concept and relation types of the language **L** when used to discuss topics in the domain **D**."

To provide common understandings in domains, logic based languages are needed for a good inference mechanism that will facilitate the reasoning on the content of the domain modelled.

Such languages are potential candidates for the representation of information for the semantic processing.

According to Laskey *et al.* (2008), modelling the uncertain aspect of the world in ontologies has attracted a lot of interest to ontology developers in the field of Artificial Intelligence (AI) especially in the World Wide Web (WWW) community. The WWW community envisions:

a) Effortless interaction between humans and computers.
b) Seamless interoperability and information exchange among web applications, and
c) Rapid and accurate identification and invocation of appropriate Web services.

As works with semantics grows more motivating, there is an increasing appreciation of the need for principled approaches in representing and reasoning under uncertainty. Uncertainty is the situation which involves imperfect and/or unknown information. The term "uncertainty" encompasses a variety of aspects of imperfect knowledge, including incompleteness, vagueness, ambiguity, and others (Laskey *et al.,* 2008).

Hence, uncertainty in ontologies needs to be tackled in order to achieve valid inferences in artificial intelligence in anticipation of the visions of WWW community.

Over the past, there has been lots of efforts made by researchers to achieve this goal, among which are fuzzy sets by Zadeh (1965) and theory of rough sets by Pawlak (1982). All these theories have their inherent difficulties as pointed out by Maji *et al.* (2001). One major problem existing in these theories is their incompatibility with the parameterization tools. To overcome these difficulties, Molodtsov (1999) initiated the concept of soft set which can be used as a generic tool for dealing with uncertainty. However, it was pointed out in Roy and Maji (2007) that classical soft set is not appropriate to deal with imprecise and fuzzy parameters. On this basis, Maji *et al.* (2001) introduced the concept of the fuzzy soft set, a more generalized concept, which is a combination of fuzzy set and soft set. In order to handle fuzzy parameters

(whose values could lie in a probable range), this research work seeks to use the concept of fuzzy soft set introduced by Maji *et al.* (2001).

1.2. Problem Statement

The lack of traditional ontological formalisms including Description Logic (DL) to support the representation of uncertainties limits them in handling and reasoning with incomplete or imprecise data in a domain.

For example, given the statement, *Musa is an intelligent student*, which could be represented in DL as:

$$\text{IntelligentStudent} \equiv \text{Student} \sqcap \text{Intelligent}$$

$$\text{IntelligentStudent(Musa)}$$

The word *intelligent* is fuzzy in nature because it is difficult to classify someone as intelligent or not intelligent in such a way that it can be accepted universally. Therefore it is difficult to establish whether the statement "*Musa is an intelligent student"* is completely true or false due to the presence of vague concept "*intelligent"*. It cannot exactly be said whether a person is beautiful or not, but rather it can only be said that a person is beautiful to some degree, hence there is the possibility of different measures to the word *intelligent* by different sources.

There are many complicated problems in economics, engineering, environment, social science, medical science, etc., that involve data which are not always all crisp (Jiang *et al.,* 2009). Therefore, problems of that nature will be common in those area. However, for an ontology to reflect the real world domain, its uncertain aspect must be modelled. This calls for a need to investigate how such vague concepts can be modelled in ontology.

1.3. Motivation

Most of the traditional tools for reasoning and computing are crisp, deterministic and precise in nature which are not likely to achieve the semantic information processing when an imprecise data is involved.

According to Keet (2014), none of the extant languages and automated reasoners that can cope with vague or uncertain knowledge have made it into mainstream Semantic Web tools.

Although, W3C shows the importance of reasoning over uncertainty in ontologies, the problem remain unsolved despite all proposed approaches which includes probability, soft set, rough set, vague set, fuzzy set theories, none of which has been standardized by W3C. Therefore there is a need to seek for the solution to this problem in order to achieve accuracy of automated reasoning in real world domain where vagueness is involved.

1.4. Aim and Objectives

The aim of this research work is to employ the use of fuzzy soft set theory in addressing the limitation of Jiang *et al.* (2010) to handle uncertainty in DL ontologies in order to be able to reason on the uncertain aspect of real world domain when fuzzy parameter are involved.

To achieve this aim, the following objectives were carried out:

a) Define a way of representing uncertainty in DL ontologies in such a way that the inference capability of the ontology are preserve.
b) Develop a reasoner that allows satisfiability reasoning on the vague aspect of DL ontologies.
c) Evaluate the proposed reasoning technique with Tableaux based and soft set ontology reasoners.

1.5. Research Methodology

In order to meet the stated objectives, the following steps were taken:

a) A deep investigation into the literatures on modelling the vague aspect of real world domain was performed.
b) A fuzzy extension to DL constructs was defined to capture the uncertain aspect of the world in DL ontologies.
c) A satisfiability reasoning technique was developed by extending the fuzzy soft set decision technique of Roy and Maji (2007) into DL reasoner.
d) The proposed reasoning technique was evaluated by using it to reason over the satisfiability of some sample DL ontologies and the results were compared with the soft set based reasoning approach and the tableaux based algorithm.

1.6. Mathematical Preliminaries

This section defines the basic mathematical notions used in this work.

1.6.1. Set

A set is a collection of objects, called the elements or members of the set. The objects could be anything: planets, squirrels, characters in Shakespeare's plays, or other sets (Hunter, 2013).

Operations on sets:

Intersection: The intersection $A \cap B$ of two sets A, B is the set of all elements that belong to both A and B; that is:

$$x \in A \cap B \text{ iff } x \in A \text{ and } x \in B.$$

Two sets A, B are said to be disjoint if $A \cap B = \emptyset$; that is, if A and B have no elements in common (Hunter, 2013).

Union: The union $A \cup B$ is the set of all elements that belong to A or B; that is $x \in A \cup B$ iff $x \in A$ or $x \in B$ (Hunter, 2013).

1.6.2. Soft Set

Molodtsov (1999) defined Soft Set in the following way: A pair *(F, E)* is called a soft set (over U) if and only if F is a mapping of E into the set of all subsets of the set U.

In other words, a soft set is a parameterized family of subsets of the set U. Every set $F(e), e \in E$, from this family may be considered as the set of e-elements of the soft set *(F, E)*, or as the set of e-approximate elements of the soft set.

1.6.3. Fuzzy Set

Zadeh (1965) defined Fuzzy Set in the following way: Let X be a space of points (objects), with a generic element of X denoted by x. Thus, $x = \{X\}$

A fuzzy set (class) A in X is characterized by a membership (characteristics) function $f_A(x)$ which associate each point in X a real number in the interval [0,1], with the value of $f(x)$ at x representing the "grade of membership" of x in A.

1.6.4. Fuzzy Soft Set

Maji, *et al.* (2001) presented the concept of the fuzzy soft sets by combining the ideas of fuzzy sets and soft set, and defined it as a mapping of each element in a set to the set of all fuzzy set of the universal set.

A pair (F, E) is called a fuzzy soft set over an initial universe U, where $F : E \rightarrow \mathcal{P}(U)$ is a mapping from E (set of parameters) into $\mathcal{P}(U)$ (set of all fuzzy sets of U).

1.7. Dissertation Organization

The remaining parts of this dissertation is organized as follows:

Chapter two reviews various literature on the state of the art of DL ontologies and their operations. A review on existing attempts of handling uncertainty in ontologies was also carried out.

In chapter three, the fuzzy soft set ontology architecture was discussed along with FSSO representation and reasoning procedures.

Chapter four, deals with the implementation of FSSO with focus on its statisfiability reasoning, it also discusses on how the FSSO reasoner differs with the tableaux based and soft set ontology reasoner.

Chapter five concludes the dissertation by summarizing the research work carried out and outlined future work.

Chapter Two

Literature Review

2.1. Introduction

This chapter briefly explains what an ontology is, the historical development of Description Logic as the basis for ontological representations. The chapter also reviews various literatures on the state of the art of DL ontologies, its operations, representation format and reasoning procedure. A review on existing attempts of handling uncertainty in ontologies is also discussed in this chapter.

2.2. Review of Ontologies

The term ontology originates from philosophy, but it has been used in many ways and across different disciplines. In the area of information science, ontology is considered as the term used to refer to the shared understanding or vocabulary of some domain of interest. An engineering viewpoint of ontology is given by Uschold and Gruninger (1996) as "an explicit account or representation of a conceptualization". This conceptualization includes a set of concepts, their definitions and their interrelationships.

Ontologies specify or model a domain using concepts, attributes, and relationships. According to Gruber (1993), ontology provides the shared vocabularies and conceptualizations of a particular domain. In general, everybody has his individual view on the world and the things he deals with on a regular basis. However, there is a common basis of understanding in terms of the language used to communicate with each other. Terms from natural language can therefore be assumed to be a shared vocabulary relying on (mostly) common understanding of certain concepts with little variety. This idea is often called conceptualization of the world. Such conceptualizations provide terminologies that can be used for communication. The main

problem with the use of a shared vocabulary according to a specific conceptualization of the world is that much of the information remains implicit. Ontologies are set out to overcome the problem of implicit and hidden knowledge by making the conceptualization of a domain explicit. The degree of explicitness of ontology greatly depends on its level of formality.

According to Smith (2003), an ontology is a dictionary of terms formulated in a canonical syntax and with commonly accepted definitions designed to yield a lexical or taxonomical framework for knowledge representation which can be shared by different information systems communities.

Ontologies have gained considerable attention and focus in research. In the present day ontologies are extensively used in different domains like knowledge engineering, artificial intelligence, natural language processing, e-commerce, intelligent information integration, information retrieval, database design and integration, bio-informatics, etc. In order to support the development of ontologies, several methodologies have been proposed to date, facilitating the process of ontology development or ontology engineering.

One of the most prominent area in which ontologies are currently being heavily deployed is the semantic web as envisioned in Berners-Lee *et al.* (2001). Tim Berners-Lee stated a decade ago that the web has well over 4.2 billion pages but the vast majority of them are in human readable format only. Consequently, software agents cannot understand and process this information, and much of the potential of the Web has so far remained untapped.

The semantic web target is that information should be retrieved from the web semantically rather than syntactically as it is been done in the current web. For this to be possible, data should be both machine readable and understandable. With this, information can be shared and processed both by automated tools, such as search engines, and by human users, enabling

intelligent information services, personalized websites, and semantically empowered search engines.

2.2.1. Historical Development of Description Logic (DL)

Research in the field of knowledge representation and reasoning is usually focused on methods for providing high-level descriptions of the world that can be effectively used to build intelligent applications. In this context, intelligent refers to the ability of a system to find implicit consequences of its explicitly represented knowledge. Such systems are therefore characterized as knowledge based systems (Nardi and Brachman, 2003).

Approaches to knowledge representation developed in the 1970's when the field enjoyed great popularity are sometimes divided roughly into two categories:

a) Logic based formalisms, which evolved out of the intuition that predicate calculus could be used unambiguously to capture facts about the world;
b) Non-logic based representations that were often developed by building on more cognitive notions. For example, network structures and rule based representations derived from experiments on recall from human memory and human execution of tasks.

Even though such approaches were often developed for specific representational chores, the resulting formalisms were usually expected to serve in general use. In other words, the non-logical systems created from very specific lines of thinking evolved to be treated as general-purpose tools, expected to be applicable in different domains and on different types of problems (Nardi and Brachman, 2003).

On the other hand, since first order logic provides very powerful and general machinery, logic based approaches were more general purpose from the very start. In a logic-based approach, the representation language is usually a variant of first order predicate calculus, and reasoning

amounts to verifying logical consequence. The non-logical approaches are often based on the use of graphical interfaces, knowledge is represented by means of some ad hoc data structures, and reasoning is accomplished by similarly ad hoc procedures that manipulate the structures.

Among these specialized representations are semantic networks and frames. Semantic Networks were developed with the goal of characterizing knowledge and reasoning by means of network shaped cognitive structures. Similar goals were shared by frame systems developed to rely upon the notion of a frame as a prototype and on the capability of expressing relationships between frames (Nardi and Brachman, 2003).

Network-based systems were often considered more appealing and more effective from a practical viewpoint than the logical systems. Unfortunately they were not fully satisfactory because of their usual lack of precise semantic characterization. The end result of this was that every system behaved differently from the others, in many cases despite virtually identical looking components and even identical relationship names. The question then arose as how to provide semantics to representation structures, in particular to semantic networks and frames, which carried the intuition that, by exploiting the notion of hierarchical structure, one could gain both in terms of ease of representation and in terms of the efficiency of reasoning.

One important step in this direction mentioned in Hayes (1979) was the recognition that frames could be given semantics by relying on first order logic. The basic elements of the representation are characterized as unary predicates, denoting sets of individuals, and binary predicates, denoting relationships between individuals. However, such a characterization does not capture the constraints of semantic networks and frames with respect to logic. Indeed, although logic is the natural basis for specifying a meaning for these structures, it turns out that frames and semantic networks did not require all the machinery of first order logic, but could be regarded as fragments of it. In addition, different features of the representation language

would lead to different fragments of first order logic. The most important consequence of this fact is the recognition that, the typical forms of reasoning used in structure based representations could be accomplished by specialized reasoning techniques, without necessarily requiring first-order logic theorem provers. Moreover, reasoning in different fragments of first order logic leads to computational problems of differing complexity.

Subsequent to this realization, research in the area of Description Logics began under the label terminological systems, to emphasize that the representation language was used to establish the basic terminology adopted in the modelled domain. Later, the emphasis was on the set of concept forming constructs admitted in the language, giving rise to the name concept languages. In more recent years, after attention was further moved towards the properties of the underlying logical systems, the term Description Logics became popular (Nardi and Brachman, 2003).

2.3. Introduction to Description Logic

This section contains definitions of Description Logic, its concept constructors, knowledge representation and reasoning. All definitions and properties stated here are obtained from Baader and Sattler (2001) and Baader *et al.* (2002).

2.3.1. Description Logic Nomenclature

Description logics (DLs) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. They are the most recent name for a family of knowledge representation formalisms that represent the knowledge of an application domain (the world) by first defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects an individual's occurring in the domain (Baader *et al.*, 2002).

Description logics are characterized by the use of various constructors to build complex concepts from simpler ones, an emphasis on the decidability of key reasoning tasks, and by the provision of sound, complete and (empirically) tractable reasoning services (Horrocks, 2002).

There exist several DLs and they are classified based on the types of constructors and axioms that they allow. The naming scheme for mainstream DLs can be summarized as: $((\boldsymbol{ALC} \mid \boldsymbol{S})\ [\boldsymbol{H}] \mid \boldsymbol{SR})\ [\boldsymbol{O}]\ [\boldsymbol{I}]\ [\boldsymbol{F} \mid \boldsymbol{N} \mid \boldsymbol{Q}]$

ALC is an abbreviation for attributive language with complements (Schmidt-Schauß and Smolka, 1991). This DL disallows RBox axioms as well as the universal role, role inverses, cardinality constraints, nominal concepts, and self-concepts.

S denotes ***ALC*** where transitivity statements are allowed, that is, roles are closed under transitivity.

Both ***ALC*** and ***S*** can be extended by role hierarchies (resulting to ***ALCH*** or ***SH***) which allow for simple role inclusions (hierarchy subsumption), e.g. $hasParent \sqsubseteq hasAncestor$.

SR denotes ***ALC*** extended with all kinds of RBox axioms as well as self-concepts.

O indicates that nominal concepts are supported, where named individuals can occur in concepts.

I indicates role inverses is allowed, where you can specify a role to be the inverse of another role.

F indicates support for role functionality. That is, each individual of the relation domain has at most one relation to an individual from the relation range

N allows for unqualified number restrictions, that is, concepts denoted by $\geq nr.\top$. Where n is a cardinality, r is the role and $\top$ is the top concept in the domain.

Q indicates support for arbitrary qualified number restrictions. (e.g. "≤ 3 hasCars.Cabriolet")

2.3.2. Description Logics Concepts Constructors

DLs are based on concepts description and roles to describe the world being modelled. Concepts, which denote the set of individuals, represent the vocabulary of an application domain. Roles denote binary relationships between individuals. Elementary descriptions are atomic concepts (unary predicate) and atomic roles (binary predicates). Complex descriptions can be built from them inductively with concept constructors. To obtain these complex descriptions, DL employs Boolean constructors such as conjunction (⊓), which is interpreted as intersection, the disjunction (⊔) which is interpreted as union and negation (¬), which is interpreted as complement, as well as the existential restriction constructor ($\exists R.C$), the value restriction constructor ($\forall R.C$), and the number restriction constructor ($\geq nR.C$), ($\leq nR.C$). Where n is a cardinality, R is a role and C is a concept (Baader *et al.*, 2002). Below are some concept constructions:

a) ⊤: denotes top concept, that is, the concept that all individuals of a domain must be instance of
b) ⊥: denote bottom concept, that is, the empty concept without no instance.
c) $\neg C$: denotes the inverse of concept C.
d) $C \sqcup D$: denotes the concept represented by the union of C and D (C 'or' D logically).
e) $C \sqcap D$: denotes C intersected with D.
f) $\exists R.C$: denotes, the set of all individuals that are in relation R to at least one individual from concept C.
g) $\forall R.C$: denotes, the set of all individuals that are in relation R with individuals from concept C.

With these operators complex concepts can be constructed from the arbitrary concepts. To use such concept expressions for axioms, two further operators are introduced: subsumption (⊑) and equivalence (≡). This allows relating concept expression to one another:

$$C \sqsubseteq D$$

$$C \equiv D$$

The interpretation of a subsumption expression like the one above is that D is a necessary condition for C (sometimes also expressed as "$D \longrightarrow C$"). Complex concept expressions can be used to define new concepts, using the equivalence operator (≡), e.g. $E \equiv C \sqcap D$.

2.3.3. Description Logic Knowledge Representation

As any knowledge representation and reasoning language, description logic language is used in a knowledge representation system to provide a language for defining a knowledge-based and tools to carry out inferences over it. Nardi and Brachman (2003) define the primary aspect involved in the realization of knowledge systems as follows:

a) Providing a precise characterization of the knowledge-based; this involves precisely characterizing the type of knowledge to be specified to the system as well as clearly defining the reasoning services the system needs to provide and the kind of questions that the system should be able to answer.
b) Providing a rich development environment where the user can benefit from different services that can make his interaction with the system more effective.

In the knowledge-base of description logic system, there is a clear distinction between intensional knowledge (TBox), or general knowledge about the problem domain, extensional knowledge (ABox), which is specific to a particular problem, and relational knowledge

(RBox), which is used to make explicit their relationships and the axioms that hold in a domain. Both TBox and ABox constitute the two main part of description logic knowledge-based.

The TBox contains intensional knowledge in the form of a terminology (hence the term TBox) and is built through declarations that describe general properties of concepts. Because of the nature of the subsumption relationships among the concepts that constitute the terminology, TBoxes are usually thought of as having a lattice-like structure; this mathematical structure is entailed by the subsumption relationship which has nothing to do with any implementation.

The ABox contains extensional knowledge, also called assertional knowledge (hence the term ABox) that is, knowledge which is specific to the individuals of the domain of discourse. Intensional knowledge is usually thought not to change. An extensional knowledge is usually thought to be contingent, or dependent on a single set of circumstances, and therefore subject to occasional or even constant change.

The RBox contains interdependencies among concepts in the domain, also called relational knowledge (hence the term RBox).

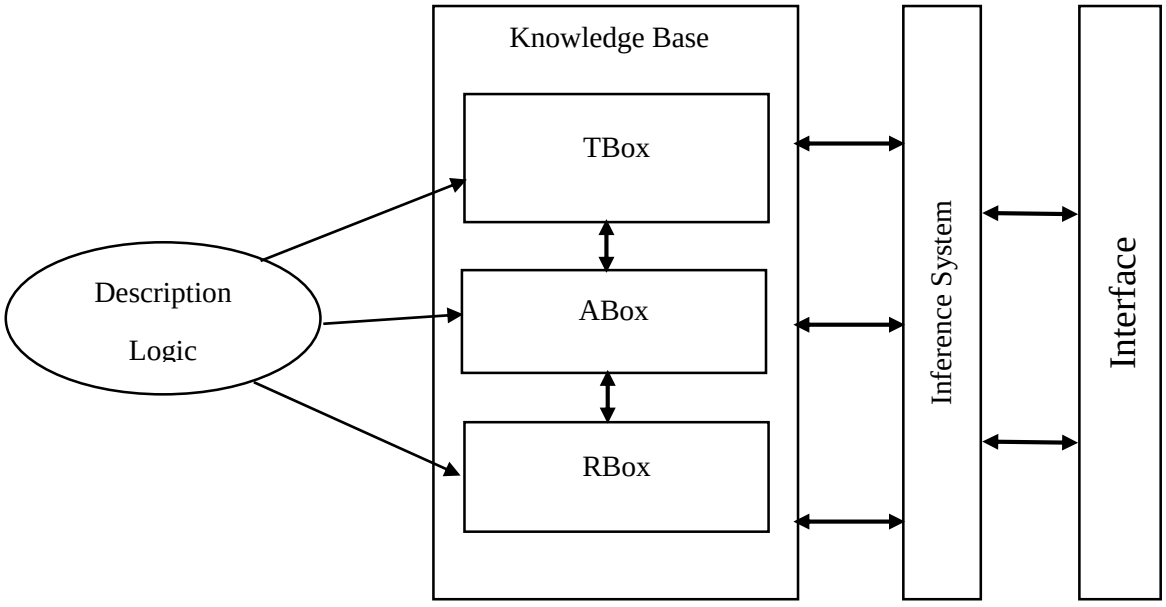

Figure 2. 1: Description logics system (Baader et al., 2007)

2.3.3.1. The Terminology Box (TBox)

Terminologies (TBoxes) are set of concepts definition or concepts inclusion, such that every concept definition or inclusion occurs at most once at the left hand side (Baader *et al.*, 2007). One key element of a DL knowledge base is given by the operations used to build the terminology. Such operations are directly related to the forms and the meaning of the declarations allowed in the TBox. The basic form of declaration in a TBox is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts. For example, a wife can be defined as a woman who has a husband who is a man by writing this declaration:

$$\text{WIFE} \equiv \text{WOMAN} \sqcap \text{has\_husband.MAN}$$

Such a declaration is usually interpreted as a logical equivalence, which amounts to providing both sufficient and necessary conditions for classifying an individual as a wife. This form of definition is much stronger than the ones used in other kinds of representations of knowledge, which typically impose only necessary conditions for classifying an individual. The strength of this kind of declaration is usually considered a characteristic feature of DL knowledge bases. In DL knowledge bases, a terminology is constituted by a set of concept definitions of the above form. There are some important common assumptions usually made about DL terminologies:

a) Only one definition for a concept name is allowed
b) Definitions are acyclic in the sense that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them.

This kind of restriction is common to many DL knowledge bases and implies that every defined concept can be expanded in a unique way into a complex expression containing only atomic

concepts by replacing every defined concept with the right hand side of its definition (Baader *et al.*, 2002).

Particularly, in concepts subsumption, the basic task in constructing a terminology is classification, which amounts to placing a new concept expression in the proper place in a taxonomic hierarchy of concepts. Classification can be accomplished by verifying the subsumption relation between each defined concept in the hierarchy and the new concept expression. The placement of the concept will be in between the most general concept that subsumes the new concept and the most specific concept that the new concept subsumes (Baader *et al.*, 2002).

2.3.3.2. The Assertional Box (ABox)

The ABox contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called membership assertions. For example, WOMAN(Jane) states that the individual *Jane* is a woman. MAN(Michael) also states that an individual *Michael* is a man.

2.3.3.3. The Role Box (RBox)

The RBox contains relational knowledge about the domain of interest, that is, interdependencies about individuals in the domain. The contents of RBox are known as role assertion and they usually appear in a binary form, implying that an instance of a concept say A is related to an instance of another concept B. For example, hasChild (Jane, John) specifies that *Jane* has as a child *John*.

2.3.4. Reasoning with Description Logics

A knowledge representation system based on DLs is able to perform specific kinds of reasoning. The purpose of a knowledge representation system goes beyond storing concept

definitions and assertions. A knowledge base comprising TBox and ABox has a semantic that makes it equivalent to a set of axioms in first order predicate logic. Thus, like any other set of axioms, it contains implicit knowledge that can be made explicit through inferences (Baader *et al.,* 2007). The different kinds of reasoning performed by a Description logic system are defined as logical inferences.

2.3.4.1. Reasoning Tasks for TBox

When a knowledge engineer models a domain, he constructs a terminology, say T, by defining new concepts, possibly in terms of others that have been already defined. During this process, it is important to find out whether a newly defined concept makes sense or whether it is contradictory. From a logical point of view, a concept makes sense if there is some interpretation that satisfies the axioms of T (that is, a model of T) such that the concept denotes a nonempty set in that interpretation. A concept with this property is said to be satisfiable with respect to T and unsatisfiable otherwise. Checking satisfiability of concepts is a key inference. In order to check whether a domain model is correct, or to optimize queries that are formulated as concepts, subsumption is used to know whether some concepts are more general than others.

A concept C is subsumed by a concept D if in every model of T, the set denoted by C is a subset of the set denoted by D. Algorithms that check subsumption are also employed to organize the concepts of a TBox in a taxonomy according to their generality.

The semantics of concept descriptions is defined in terms of an interpretation $I = (\Delta^I, \bullet^I)$, consisting of a domain of interpretation (also known as domain of individuals) Δ^I which is a non-empty set of individuals, and an interpretation function $\bullet^I$ mapping every atomic concept A to a set $A^I \subseteq \Delta^I$ and every atomic role r to a binary relation $r^I \subseteq \Delta^I \times \Delta^I$. The extension of $\bullet^I$ to arbitrary concept descriptions is inductively defined as shown below.

Table 2. 1: Syntax and semantics of concept descriptions (Baader and Sattler, 2001)

Constructor	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
at-least restriction	$(\geqslant nr.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
at-most restriction	$(\leqslant nr.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

The concept description D subsumes the concept description C (written $C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations *I*; C is satisfiable iff there exists an interpretation *I* such that $C^I \neq \emptyset$; C and D are equivalent iff $C \sqsubseteq D$ and $D \sqsubseteq C$; and C and D are disjoint iff $C_I \cap D_I = \emptyset$ for all interpretations *I.*

Traditionally, the basic reasoning mechanism provided by DL systems check the subsumption of concepts. This, in fact, is sufficient to implement also other inferences since they can be reduced to subsumption problem as can be seen by the following reductions. For concepts C and D, it implies that:

a) C is unsatisfiable if, C is subsumed by⊥;

b) C and D are equivalent, if, C is subsumed by D and D is subsumed by C;

c) C and D are disjoint, if, $C \sqcap D$ is subsumed by⊥.

All description languages implemented in actual DL systems provide the intersection operator "⊓" and almost all of them contain an unsatisfiable concept. Thus, most DL systems that can

check subsumption can perform all four inferences in definition 1. If, in addition to intersection, a system allows one also to form the negation of a description, one can reduce subsumption, equivalence, and disjointness of concepts to the satisfiability problem. This can be done through the following rules:

a) C is subsumed by D, iff $C \sqcap \neg D$ is unsatisfiable;

b) C and D are equivalent, iff both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;

c) C and D are disjoint, iff $C \sqcap D$ is unsatisfiable.

2.3.4.2. Reasoning Tasks for ABoxes

After a knowledge engineer has designed a terminology and has used the reasoning services of her DL system to check that all concepts are satisfiable and that the expected subsumption relationships hold, the ABox can be filled with assertions about individuals. It was shown in section 2.2.3.2 and 2.2.3.3 that there are two kinds of assertions, concept assertions of the form C(a) and role assertions of the form r(a, b). Of course, the representation of such knowledge has to be consistent, because otherwise, from the viewpoint of logic, one could draw arbitrary conclusions from it.

Let N_I be a set of individual names. An ABox is a finite set of assertions of the form C(a) (concept assertion) or r(a, b) (role assertion), where *C* is a concept description, *r* a role name, and a, b are individual names.

An interpretation *I*, which additionally assigns elements $a^I \in \Delta^I$ to individual names *a*, is a model of an ABox *A* iff $a^I \in C^I\big((a^I, b^I) \in r^I\big)$ holds for all assertions $C(a)\big(r(a,b)\big)$ in *A*. The Abox *A* is consistent iff it has a model. The individual *a* is an instance of the description *C* w.r.t. *A* iff $a^I \in C^I$ holds for all models *I* of *A*.

The expansion of A with respect to T as the ABox is defined as A' that is obtained from A by replacing each concept assertion C(a) in A with the assertion $C'(a)$, where C' is the expansion of C with respect to T. In every model of T, a concept C and its expansion C' are interpreted in the same way. Therefore, A' is consistent with respect to T iff A is consistent. However, since A' does not contain a name symbol defined in T, it is consistent with respect to T iff T is consistent. It can then be concluded that A is consistent with respect to T iff its expansion A' is consistent.

Satisfiability of a concept descriptions as well as the instance problem can be reduced to the consistency problem for ABoxes:

a) C is satisfiable iff the ABox $\{C(a)\}$ for some $a \in N_I$ is consistent; and

b) a is an instance of C w.r.t. A iff $A \cup \{\neg C(a)\}$ is inconsistent.

If a knowledge base is considered as a means to store information about individuals, it may be desired to know all individuals that are instances of a given concept description C, that is, the description language is used to formulate queries. The retrieval problem is, given an ABox A and a concept C, to find all individuals a, such that A is entailed by C(a). A non-optimized algorithm for a retrieval query can be realized by testing for each individual occurring in the ABox whether it is an instance of the query concept C.

2.3.4.3. The Tableaux-Based Algorithm

Given an arbitrary *DL*-concept C, the tableaux algorithm for satisfiability tries to construct a finite interpretation I that satisfies C, i.e., contains an element x such that $x \in C$. However, if we look at the information that must be expressed (namely, the elements of the interpretation, the concept descriptions they belong to, and their role relationships), it will be seen that ABox assertions are sufficient for this purpose. It will be convenient to assume that all concept descriptions are in Negation Normal Form (NNF), i.e., negation occurs only directly in front

of concept names. Using de Morgan's rules and the usual rules for quantifiers, any concept definition can easily be transformed to an equivalent one in NNF by pushing negations inwards using the following equivalences:

a) $\neg\,(A \sqcup B) = \neg A \;\sqcap\; \neg B$

b) $\neg\,(A \sqcap B) = \neg A \sqcup \neg B$

c) $\neg(\exists r.C) = \;(\forall r.\neg C\,)$

d) $\neg(\forall r.C\,) \;= (\exists r.\neg C)$

e) $\neg(\leq n\, r.C) \;=\geq (n+1)\, r.C)$

f) $\neg(\geq n\, r.C) = \leq (n-1)\, r.C$

The Tableaux algorithm uses four rules to come to a decision: intersection elimination, union elimination, existential elimination and universal elimination. The goal is considered the label of the root node of a proof tree. Then, application of the proof rules transform the node, add elements to its label set, and spawn new nodes expanding the tree structure where the application of rules is repeated. This is continued until no more rules can be applied to any of the nodes in the tree. At this point, the proof tree (or the "tableaux") is said to be saturated. If any of the branches of the tree contains an obvious contradiction, that is an expression C and its negation $\neg C$, this branch is said to be closed. Branches where this is not the case are called open. The initial goal of the proof is satisfiable iff the tree does not contain any closed branches.

The first two rules, intersection elimination and union elimination (also *and*- and *or*-elimination), operate directly on the formulae of the current node. The application of the intersection rule adds the two operands to the node's label set, i.e. to the set of DL expressions that are associated with this node. Union connectives allow the addition of their first operand, and if the whole tree fails, the addition of their second operand is tried as an alternative. The following short hand gives a first impression of the two rules:

a) $x : \{...C \sqcap D \ ...\} \mapsto x : \{...C, D...\}$

b) $x : \{...C \sqcup D...\} \mapsto x : \{...C, ...\}\ or\ x : \{...D, ...\}$

x identifies the current node, the list in braces denotes its node label, consisting of various Description Logics expressions. In the first expression, the node label contains an intersection. Applying the intersection rule transforms the node label to the expression at the right of the arrow: the two operands of the intersection are added as individual members to the node label. Similarly, in the second expression the union rule adds its first or second operand to the node's label.

The existential elimination creates an edge and a new node, thereby extending the proof tree. The edge is labelled with the role name that was governed by the existential. The restricting concept of the existential is carried over to the new node. The universal elimination depends on an existing edge which is labelled with the name of the role governed by the universal. Then the restricting concept of the universal is carried over to the corresponding node, if this node does not contain it already. In short, for nodes *x* and *y* and an edge labelled with R, the result of the two operations would be:

a) $x : \{...\exists R.C...\} \mapsto x : \{...\} — R — y : \{C\}$

b) $x : \{...\forall R.C...\} — R — y : \{...\} \mapsto x : \{...\} — R — y : \{..., C\}$

In the first operation, the node *x* contains an existential expression in its node label. Applying existential elimination (shown to the right of the arrow), the node gets a child node *y*, connected by an edge which is labelled with the name of the existential's relation R. The existential's concept, C, is the initial contents of *y*'s node label. In the second operation, *x* contains a universal expression, and is additionally connected to a child node *y* through an edge that has the same name as the universal's relation. Then universal elimination can be applied, leading

to the situation to the right of the arrow, where the universal's concept expression C is added to the child's node label.

The process is carried on, until there are no more rules to apply (i.e. until the tableaux is saturated). If a contradiction shows up on any of the branches of the tree, listing both a concept and its complement in the same node, that closes the tableaux and proves the unsatisfiability of the goal. If this is not the case, then there is a satisfying model for the goal.

Appropriate conclusions have to be drawn to answer the initial hypothesis. For instance, in the above example, $C \sqsubseteq D$ would be true, if $C \sqcap \neg D$ is unsatisfiable. Because, the hypothesis $C \sqsubseteq D$ leads to the goal $C \sqcap \neg D$, since a concept C is subsumed by another concept D iff the intersection of C with the complement of D is empty. The goal can be easily tested for satisfiability, i.e. it can be tested whether the intersection between the two concepts is empty or non-empty. This is difficult with the first expression (which would require to show that each individual in C is necessarily also in D, in order to prove the hypothesis). But falsifying (i.e. proving the non-satsifiability of) the second expression (the goal) proves the satisfiability of the first (the hypothesis). The complete list of rules is given in Figure 2.3 below,

$\sqcap -rule$ *if* 1. $(C_1 \sqcap C_2) \in L(x)$

2. $\{C_1, C_2\} \notin L(x)$

then $L(x) \longrightarrow L(x) \cup \{C_1, C_2\}$

$\sqcup -rule$ *if* 1. $(C_1 \sqcup C_2) \in L(x)$

2. $\{C_1, C_2\} \cap L(x) = \emptyset$

then a. save **T**

b. $tryL(x) \longrightarrow L(x) \cup \{C_1\}$

If that leads to a clash then restore **T** and

c. $tryL(x) \longrightarrow L(x) \cup \{C_2\}$

$\exists - rule$ *if* 1. $\exists R.C \in L(x)$

2. there is no *y* such that $L(\langle x, y\rangle) = R\ and\ C \in L(y)$

then create a new node *y* and edge $\langle x, y\rangle$ with $L(y) = \{C\}$ and $L(\langle x, y\rangle) = R$

$\forall - rule$ *if* 1. $\forall R.C \in L(x)$

2. there is some *y* such that $L(\langle x, y\rangle) = R\ and\ C \notin L(y)$

then $L(y) \longrightarrow L(y) \cup \{C\}$

Figure 2. 2: Tableaux Inference Rule for ALC (Horrocks, 1997)

Where:

a) C, C_1 and C_2 are arbitrary DL concepts

b) R is a relation

c) **T** denotes the whole proof tree

d) *x* and *y* denote specific nodes in the tree

e) $L(x)$ signifies the node label of node *x*, i.e., the set of DL formulae associated with this node.

2.3.4.4. The Soft Set Ontology Decision Making Procedure

Given a DL-knowledge base, Soft Set Ontology SSO can be defined as

$$SSO = \langle RB_S, TB_S, AB_S\rangle$$

Where: RB_S, TB_S, and AB_S are the RBox, TBox, and ABox of SSO respectively.

The interpretation I for SSO can be obtained according to the semantics of DLs if $I \vDash RB_S$, $I \vDash TB_S$ and $I \vDash AB_S$, then we have that *I* is a model of SSO. Assume that the set of decision parameters $M = \{C_1, C_2, \ldots, C_n\}$. Hence, for each parameter $C_i \in M, i \in \{1, 2, \ldots, n\}$, we have $C_i^I \subseteq \Delta^I$. Thus, an ontology-based soft set is obtained as follows:

$$\langle I, M\rangle = \{(C_1, C_1^I), (C_2, C_2^I), \ldots, (C_n, C_n^I)\}$$

The ontology-based soft set semantic decision making algorithm for a query is shown below.

Input: An ontology-based soft set $\langle I, M\rangle = \{(C_1, C_1^I), (C_2, C_2^I), \ldots, (C_n, C_n^I)\}$, a soft set ontology SSO, a domain ontology DO, and a query $Q = \{Q_1, Q_2, \ldots, Q_k\}$ $(k \leq n)$.

Output: The optimal alternative.

Step 1: Obtain an ontology-based soft set $\langle J, N\rangle$ for the query Q in the following way: for each query parameter $Q_i \in Q$ $(1 \leq i \leq k)$.

(1) If there exists a decision parameter C_j of $\langle I, M\rangle$ $(1 \leq j \leq n)$ such that $\{SSO, DO\} \vDash Q_i \equiv C_j$, then we obtain an element (Q_i, C_j^I) of $\langle J, N\rangle$. Goto (4).

(2) If there exists a decision parameter C_j of $\langle I, M\rangle$ $(1 \leq j \leq n)$ such that $\{SSO, DO\} \vDash Q_i \sqsupseteq C_j$, then we obtain an element (Q_i, C_j^I) of $\langle J, N\rangle$. Goto (4).

(3) If there exists a decision parameter C_j of $\langle I, M\rangle$ $(1 \leq j \leq n)$ such that $\{SSO, DO\} \vDash Q_i \sqsubseteq C_j$, then we obtain an element (Q_i, C_j^I) of $\langle J, N\rangle$. Goto (4).

(4) If any query parameter $Q_i \in Q$ $(1 \leq i \leq k)$ has Q_i-elements (or Q_i-individuals), then Goto (5); else, that is, there exists one query parameter $Q_i \in Q$ $(1 \leq i \leq k)$ such that Q_i has not Q_i-elements (or Q_i-individuals), then output ‘‘Cannot make decisions and please input different decision making parameters’’ and Goto (5).

(5) Obtain the ontology-based soft set $\langle J, N\rangle = \{(Q_1, C_1^I), (Q_2, C_2^I), \ldots, (Q_k, C_k^I)\}$.

Step 2: Present the ontology-based soft set $\langle J, N\rangle$ in tabular form and compute the choice value c_i of o_i, $\forall i$.

Step 3: The optimal decision is to select o_l if $c_l = max_i\, c_i$ and output o_l.

Step 4: If l has more than one value then any one of o_l may be chosen.

Algorithm 2. 1: Ontology-based soft set semantic decision algorithm (Jiang et al., 2010)

2.4. Modelling Uncertainty in Ontology

The traditional ontologies assume crisp (true or false) meaning for concepts and relations in a domain and do not take into consideration the degree of uncertainty inherent to some concepts in the domain. Therefore, interpretation with such uncertainty concepts will not always be satisfiable even in most expressive DLs, as illustrated in Figure 2.3.

In a real world domain imprecise (uncertain) concepts are inevitable. Imprecise here represents the lack of having a full knowledge about objects in the domain. It is intended to cover a variety of forms of incomplete knowledge, including incompleteness, vagueness, ambiguity, and others.

In Chapter Three, uncertainty was handled by extending vague concepts with fuzzy values. The obtained model called Fuzzy Soft Set Ontology (FSSO) is able to handle both crisp and the vague aspect of the world domain. The approach adopted is attaching a degree of membership to the vague concepts in the domain through the ABox and leaving the TBox unmodified since the TBox is the core knowledge of an ontology knowledge base. The detailed discussion of the proposed FSSO is discussed in Chapter Three.

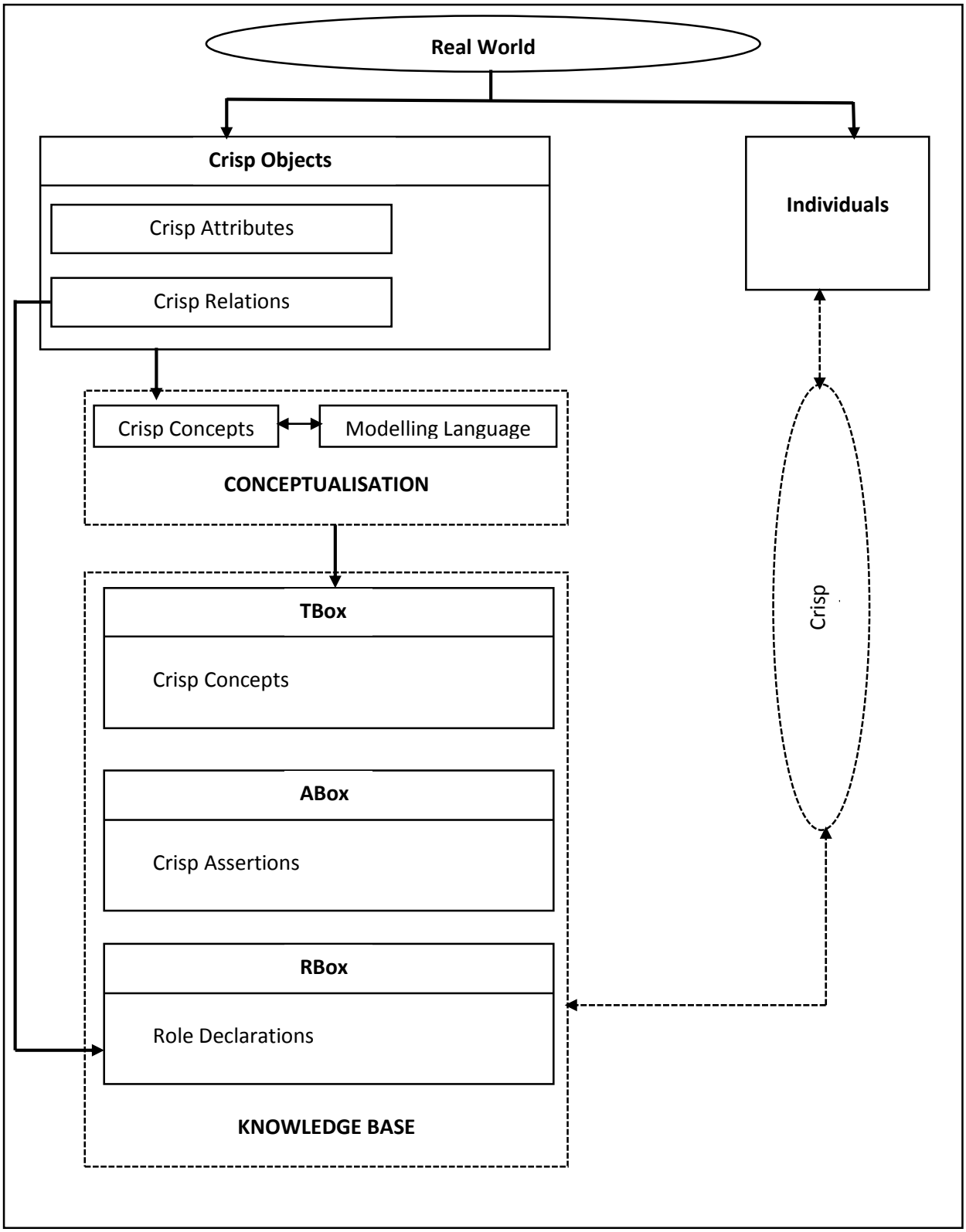

Figure 2. 3: Ontologies Modelling (Kana and Akinkumi, 2015)

2.5. Related Works

Modelling uncertainty in ontologies is a concern of ontology developers especially in the WWW community. The need to model and reason with uncertainty has been found in many different semantic web contexts such as matchmaking in web services, classification of genes in bioinformatics, multimedia annotation and ontology learning. In an effort to handle uncertainty in the web, the W3C founded the uncertainty reasoning for the World Wide Web Incubator Group (Laskey *et al.*, 2008). In their final report, they presented requirements for better defining the challenge of reasoning with and representing uncertain information available through the WWW and related WWW technologies. The report focuses on the following points:

a) Identifying and describing situations on the scale of the World Wide Web for which uncertainty reasoning would significantly increase the potential for extracting useful information.
b) Identifying methodologies that can be applied to these situations and the fundamentals of a standardized representation that could serve as the basis for information exchange necessary for these methodologies to be effectively used.
c) Including a set of use cases illustrating conditions under which uncertainty reasoning is important.
d) Providing an overview and discusses the applicability to the World Wide Web of prominent uncertainty reasoning techniques and the information that needs to be represented for effective uncertainty reasoning to be possible.
e) Including a bibliography of work relevant to the challenge of developing standardized representations for uncertainty and exploiting them in Web-based services and applications.

However, as noted by Laskey *et al.* (2008), W3C is yet to achieve anything in this direction.

Straccia (2001) focused on SHOIN (D), and added fuzzy to SHOIN (D) which was shown to have more representation and reasoning capabilities beyond classical SHOIN (D), the classical SHOIN (D) allows to reason with concrete data types, such as strings and integers using so-called concrete domains but in their own approach, concrete domains was based on fuzzy sets as well. A concrete fuzzy domain as they defined it, is a pair $\langle \Delta_D, \Phi_D \rangle$, where Δ_D is an interpretation domain and Φ_D is the set of concrete fuzzy domain predicates d with a predefined arity n and an interpretation $d^D : \Delta_D^n \rightarrow [0,1]$, which is a n-ary fuzzy relation over Δ_D.

BayesOWL is a probabilistic framework for modelling uncertainty in semantic web (Pan *et al.*, 2005). In their approach, they categorized ontologies into source and target ontologies and then translated them into Bayesian networks (BN); the mapping concept between the two ontologies are treated as evidential reasoning between the two translated BNs. Probabilities needed for constructing Conditional Probability Tables (CPT) during translation and for measuring semantic similarity during mapping are learned using text classification techniques where each concept in an ontology is associated with a set of semantically relevant text documents, which are obtained by ontology guided web mining.

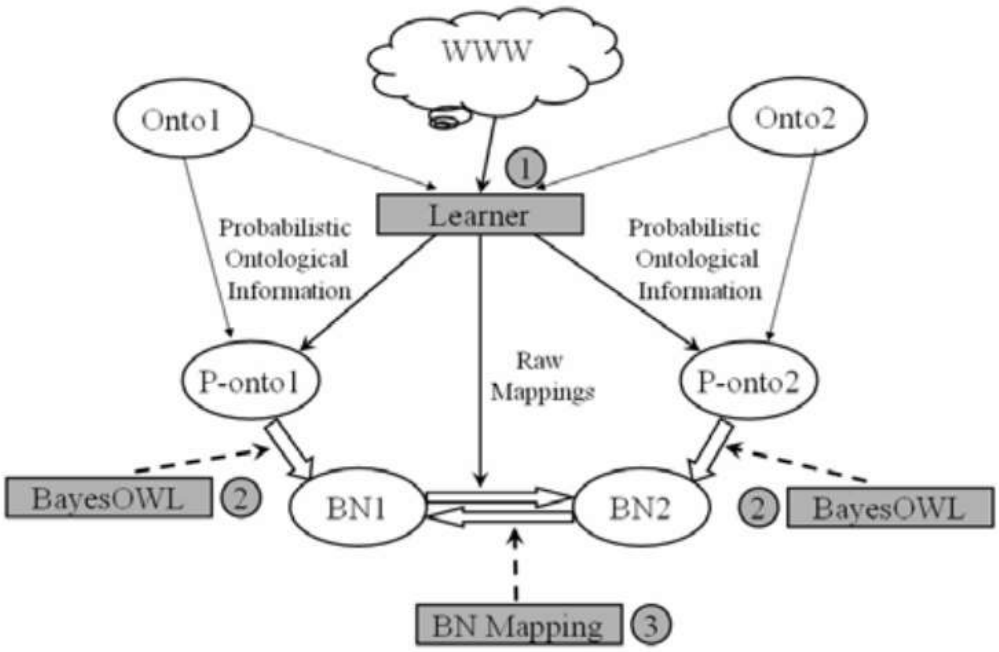

Figure 2. 4: BayesOWL Framework (Pan et al., 2005)

Sanchez and Tettamanzi (2006) introduced fuzzy description logic with extended qualified quantification called $ALCQ_F^+$, which allows for the definition of fuzzy quantifiers of the absolute and relative kind by means of piecewise linear functions on $\mathbb{N}$ and $\mathbb{Q} \cap [0,1]$ respectively. These quantifiers extends the usual (qualified) $\exists$, $\forall$ and number restriction. Their reasoning algorithm calculates the satisfiability interval for a fuzzy concept.

Costa and Laskey (2006) proposed an extension to OWL called PR-OWL which is a framework for probabilistic ontologies, by making a formal definition of a probabilistic ontology. The extension, PR-OWL, is based on Multi-Entity Bayesian Networks (MEBN), a first-order Bayesian logic that unifies Bayesian probability with First-Order Logic. PR-OWL combines the full representation power of OWL with the flexibility and inferential power of Bayesian logic.

Stoilos *et al.* (2007) made a fuzzy extension of SHIN DL to ALC DL, they focus on improving the expressiveness of ALC by allowing SHIN constructors such as: transitive role axioms (S), inverse roles (I), role hierarchies (H) and number restrictions (N).

Roy and Maji (2007) used an application of fuzzy soft set theory in object recognition problem. The recognition strategy is based on multi-observer input parameter data set. The algorithm involves construction of comparison table from the resultant fuzzy soft set and the final decision is taken based on the maximum score computed from the comparison table.

Pronto is a non-monotonic probabilistic reasoner for expressive DLs which is capable of processing knowledge bases containing about a thousand of probabilistic axioms (Klinov and Parsia, 2008). Pronto was presented in order to perform a probabilistic reasoning in the semantic web. Pronto reason about uncertainty in OWL ontologies by establishing the probabilistic relationships between OWL classes and probabilistic relationship between an OWL class and an individual. One of the principal requirements for pronto was that the uncertainty could be introduced into OWL ontologies and that existing OWL reasoning services should be retained. To meet that requirement, Pronto was designed on top of the OWL reasoner to provide routines for higher level probabilistic reasoning procedures. One attractive feature of these probabilistic logics is that they allow modelers to declaratively describe their uncertain knowledge without fully specifying any probability distribution (in contrast to Bayesian networks).

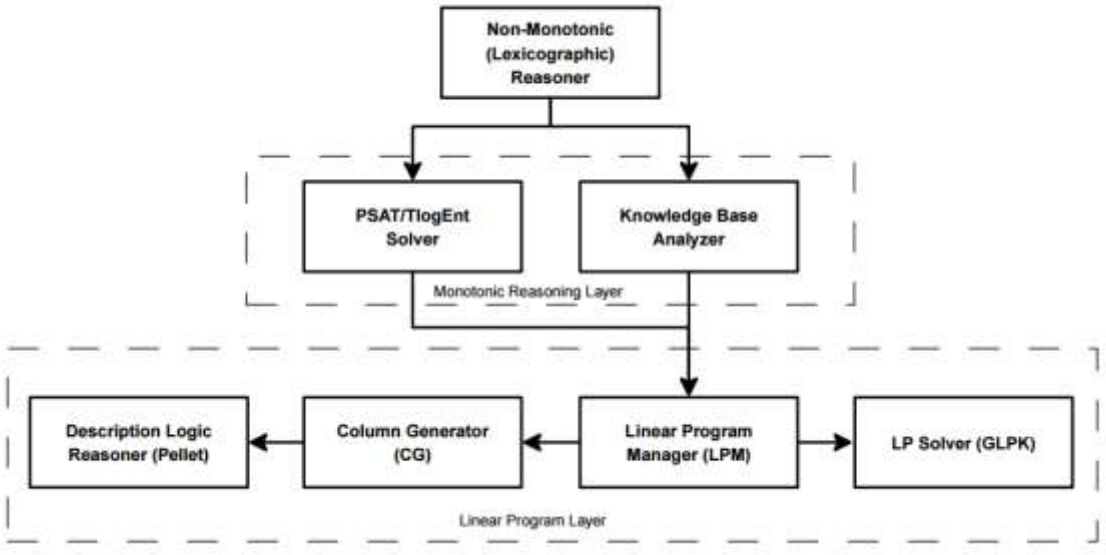

Figure 2. 5: Architecture of Pronto (Klinov and Parsia 2008)

Kong *et al.* (2009) noted that the algorithm for the selection of optimal object in Roy and Maji (2007) is incorrect. They gave a counter-example to illustrate that using the algorithm the right choice cannot be obtained in general. So they proposed another algorithm based on maximum choice value as the basis for selection of optimal object.

Jiang *et al.* (2009) presented an extended soft set theory by using the concepts of Description Logics DLs to act as the parameters of soft sets, they defined some operations for the extended soft sets such as: equality of two extended soft sets, subset and superset of an extended soft set, complement and restricted difference of an extended soft set, union and restricted intersection, restricted union and extended intersection of two extended soft set. They also prove that certain De Morgan's law hold in the extended soft set theory with respect to these operations they defined.

Zhao (2010) surveyed existing approaches in uncertainty extensions to DLs and ontologies, they noted that Semantic Web applications had revealed a shortcoming of standard Description Logics, i.e., their limited knowledge expressivity, they reviewed earlier approaches used by researcher in handling uncertainty most of which are either probabilistic or fuzzy logic.

Knowing fully-well that Description Logics (DLs) and Logic Programs (LP) are the two main categories of logic-oriented knowledge representation formalisms for the Semantic Web, they consider researches that attempted to extends terminological concepts defined in a DL theory by means of rules defined in a LP theory, but since the LP they considered is limited to Horn Logic expressivity, they found out that the naive combination of even a very simple DL with an arbitrary Horn Logic program is undecidable as shown by Levy and Rousset (1998).

Jiang *et al.* (2010a) proposed an approach on semantic decision making using ontology-based soft set. They pointed out that the traditional approaches to soft set based decision making are not fit to solve decision making problem involving user queries and then presented an approach to semantic decision making by using ontology-based soft sets and DL reasoning, and lastly proposed an algorithm for implementing semantic decision making.

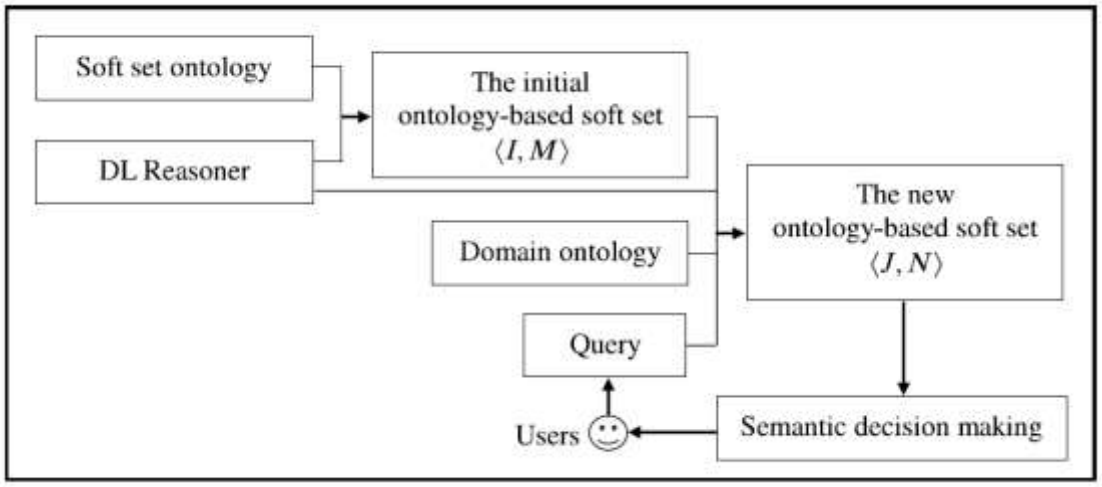

Figure 2. 6: Architectural implementation of ontology-based soft set (Jiang et al., 2010)

Feng *et al.* (2010) upheld Roy and Maji (2007) method of using maximum scores for selecting optimal decision as a general useful method and concluded that the counterexample given by Kong *et al.* (2009) is based on an improper understanding of maximum choice value as the

optimal decision; hence it is not sufficient for showing that the Roy and Maji (2007) method is incorrect, but then they noted some inherent limitations of Roy and Maji (2007) as:

a) It could hardly arrive at the final optimal decision when multiple objects have the same scores (the algorithm only choses any among such objects).
b) Large computation complexity when dealing with large parameter set or great number of objects.

As such they presented a different approach to fuzzy soft set decision making methodology by using level soft sets, this is done by considering different types of thresholds of the soft set (such as t-level soft sets, mid-level soft sets and top-level soft sets), which could yield different final optimal decisions based on different level soft sets. Therefore the proposed approach is an adjustable method which captures an important feature for decision making in an imprecise environment.

Jiang *et al.* (2010b) presented a combination of an interval-valued intuitionistic fuzzy set theory and a soft set theory. Their theory draw across definitions of Interval-Valued Intuitionistic Fuzzy Set (IVIFS) by Atanassov and Gargov (1989) which is primarily characterized as interval-valued membership degree and interval-valued non-membership degree, then making a set a parameterized family of interval-valued intuitionistic fuzzy subsets of a set say U, thus it is still a mapping from parameters to $IVIFS(U)$.

Bellenger and Gatepaille (2011) proposed an approach of addressing the problem of representing uncertainty based on the Dempster-Shafer theory. They constructed a Dempster-Shafer ontology that can be imported into any specific domain ontology and to instantiate it in an uncertain manner. To reason about the uncertain objects contained in the ontology, they implemented a Java application to retrieve the instances associated to each individual uncertain

concept, the Dempster-Shafer information is collected and then transmitted to a common basic Dempster-Shafer library which then performs calculations in the Dempster-Shafer theory.

BeliefOWL was proposed by Essaid and Yaghlane (2011), it is an approach for extending an OWL ontology with belief functions and focuses on translating an ontology into a directed evidential network by applying a set of structural translation rules. Once the network is constructed, belief masses will be assigned to the different nodes in order to propagate uncertainties later. This approach is also modelled via the Dempster-Shafer theory of evidence.

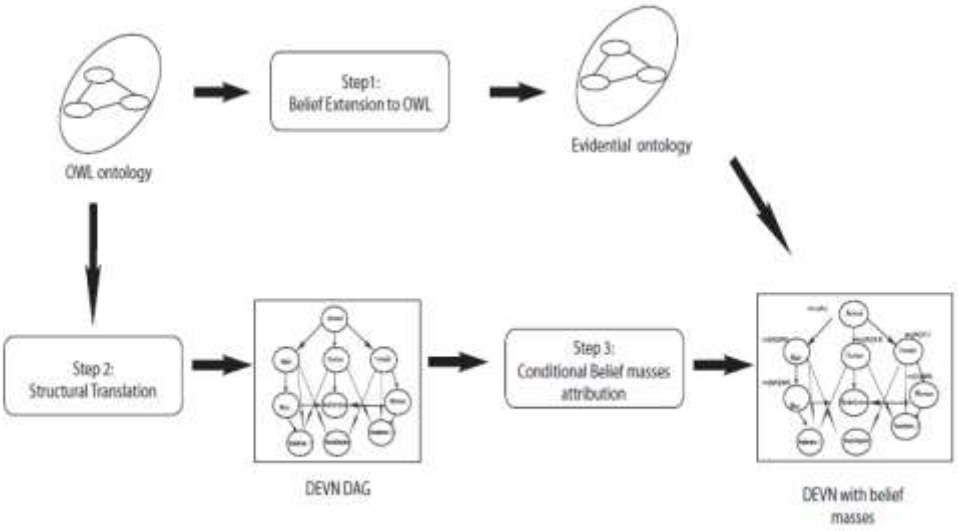

Figure 2. 7: BeliefOWL Framework (Essaid and Yaghlane, 2011)

Cagman *et al.* (2011) redefined fuzzy soft set constructs and properties owning to the limitation and incorrectness claims by Kong *et al.* (2009) on the algorithm in Roy and Maji (2007). They defined fuzzy soft aggregation operator that produces an aggregate fuzzy set from a fuzzy soft set and its cardinal set. The approximate functions of a fuzzy soft set are fuzzy. The aggregation operator on the fuzzy sets is an operation by which several approximate functions of a fuzzy soft set are combined to produce a single fuzzy set which is the aggregate fuzzy set of the fuzzy soft set. They showed that once an aggregate fuzzy set has been arrived at, it may be necessary to choose the best single crisp alternative from this set.

Cheng *et al.* (2013) presented an algorithm for multi-attribute decision making based on the optimum object for ideal and spatial distance. The algorithm considers the difference of the alternative decisions completely by computing the distance between each optimal objects in the fuzzy soft set and the optimum object for ideal points in a set, and then more rational decision making result can be obtained from the minimum distance calculated.

Kana and Akinkunmi (2014) defined a way of instantiating ontologies of vague domains using the concept of soft sets initiated by Molodtsov (1999) and the concept of rough sets introduced by Pawlack (1982). They defined ontological algebraic operations and their properties while taking into consideration the uncertain nature of domains. They showed that, by doing so, intra ontological operations and their properties are preserved and formalized as operations in a vague set of objects and can be proved algebraically.

Kana and Akinkumi (2016) showed that the source of vagueness in an ontology is from vague attributes and vague roles. They split the set of roles R and set of attributes A into four distinct sets R_C, R_V, A_C and A_V representing the set of crisp and vague roles and attributes respectively to have a clear separation between crisp and vague concepts depending on whether vague attributes or roles are used in the conceptualization.

2.6. Gap in the Literature

Approaches studied in the literature only focused on handling uncertainty with soft set, fuzzy set, vague set, rough set theories or some probability concepts, none of the approach in the literature survey use fuzzy soft set theory to handle the problem of uncertainty in ontology context. The nearest work with the fuzzy soft set are those literature that tried using fuzzy soft set for decision making but certainly it was not extended to ontology. Whereas, there was an approach of semantic decision making using ontology-based soft set by Jiang *et al.* (2010) it was noted that this approach is not appropriate to deal with imprecise and fuzzy parameters

(i.e, parameters which the boundaries of application can vary considerably according to context or conditions, instead of being fixed once and for all, example; Temperature could be “hot” and “cold” in a particular context while at other context it could be “high”, “medium” or “low”).

Chapter Three

Fuzzy Soft Set Ontology

3.1. Fuzzy Soft Set Ontology Architecture

The proposed approach to model uncertainty in DL ontologies extends crisp concepts shown in Figure 2.4 with fuzzification component to represent fuzzy concepts in a domain of interest, which then resulted to a transformed ontology called Fuzzy Soft Set Ontology (FSSO) that handles uncertainty by attaching a degree of membership to vague concepts of a domain. The obtained FSSO is then passed to a reasoner together with a user query (user query must be any concept name out of the concepts represented in TBox), the reasoner does the following:

(1) Expand the concepts used in the user query up to their terminal concepts with the normal Tableaux expansion procedure. (Terminal concepts are concepts not defined in terms of any other concept, and are used to build more complex concepts in a domain)

(2) Search in the ABox of FSSO and extract the value for the terminal concepts with regards to the instance passed in the user query or all instances in the domain if there was no any instance passed in the user query.

(3) Construct a comparison table with instances in the domain as the rows and the query terminal concepts as the columns of the table with the values extracted in (2) above as the corresponding entries of the table.

After the construction of a comparison table, an optimal solution can be obtained by using the interpretation semantics of the constructors involved in the expansion of the query parameter. Hence the uncertainty of the concept(s) involved is resolved among all the instances in the domain. Figure 3.1 illustrates the FSSO architecture pictorially.

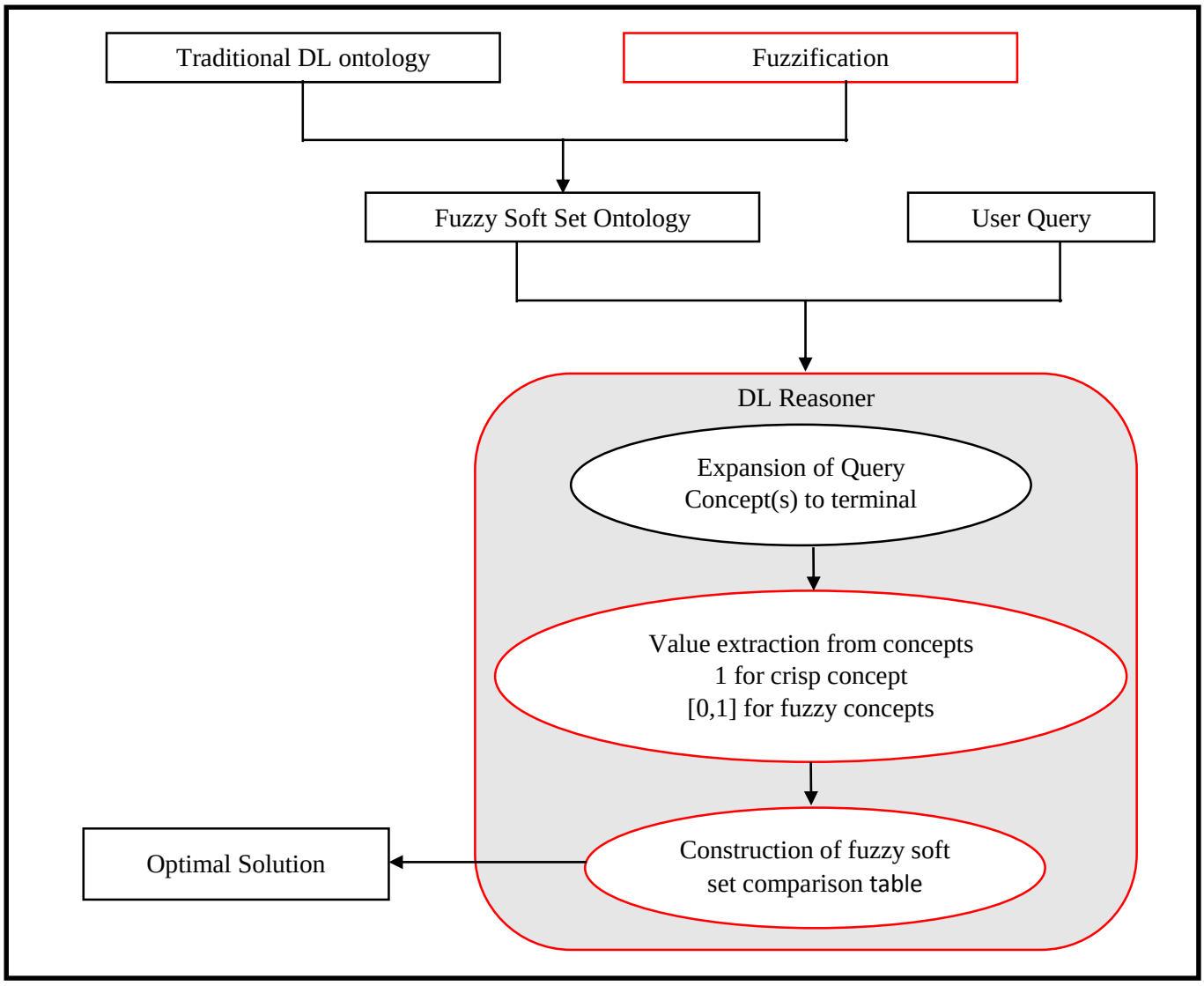

Figure 3. 1: Proposed fuzzy soft set ontology architecture

This proposed architecture is designed to address the limitation of (Jiang *et al.*, 2010), which lack the reasoning capability for imprecise concepts that could lie in a probable range [0,1].

3.1.1. Concept/Instance Fuzzification

In this approach, for each assertional statement of a fuzzy concept, the fuzzy membership values is evaluated through a mapping function $c: I \rightarrow [0,1]$, where $c \in C$ (set of concepts in the domain), $I = \{i_1, i_2, i_3, \dots, i_n\}$ is a finite set of instance and [0,1] is a fuzzy value in the range of 0 and 1. Fuzzy values are determined by the ontology builder and/or expert of a domain

of discourse using the knowledge acquisition techniques acquired over time such as data mining. For example, let assume a concept pretty belongs to a set of concepts in a domain of discourse, assuming Mary is also an instance in the domain. So to make the assertion "Mary is pretty" will require the fuzzy mapping function pretty to assign a fuzzy value to the assertion "Mary is pretty" since the concept pretty is fuzzy in nature. Assuming the ontology builder has determined Mary is pretty to 0.7 degree then the assertion will be represented as $Pretty(Mary, 0.7)$.

3.2. Representing Fuzzy Soft Set Ontology

Given a fuzzy DL-knowledge base, Fuzzy Soft Set Ontology (FSSO) can be represented as a triple $FSSO = \langle RB_F, TB_F, AB_F \rangle$, where RB_F, TB_F, and AB_F are the RBox, TBox, and ABox of FSSO respectively.

The interpretation I for FSSO is obtained by combining (1) Set of decision parameters $M = \{m_1, m_2, \ldots, m_n\}$ representing terminal concepts in the domain. (2) Domain of Interpretation Δ^I which is the set of all instances represented in the FSSO. The combination will then results to:

$$\langle I, M \rangle = \{(m_1, m_1^I), (m_2, m_2^I), \ldots, (m_n, m_n^I)\}.$$

Where, $m_i \in M$, $i \in \{1, 2, \ldots, n\}$ are the decision parameters and m_i^I is the set of instances that satisfied the parameter m_i. Hence, the interpretation of each m_i is a proper subset of domain of interpretation ($m_i^I \subseteq \Delta^I$) .

A DL ontology of family domain was extended with vague concepts to form a fuzzy soft set ontology. The $FSSO = \langle RB_F, TB_F, AB_F \rangle$ is defined as follows:

$$RB_F = \emptyset$$

$$TB_F = \{$$

$$Woman \equiv Person \sqcap Female$$

$$Man \equiv Person \sqcap \neg Female$$

$$Mother \equiv Woman \sqcap \exists hasChild.Person$$

$$Father \equiv Man \sqcap \exists hasChild.Person$$

$$Parent \equiv Father \sqcup Mother$$

$$Grandmother \equiv Mother \sqcap \exists hasChild.Parent$$

$$MotherWithManyChildren \equiv Mother \sqcap \geq 3 hasChild$$

$$MotherWithoutDaughter \equiv Mother \sqcap \forall hasChild.\neg Female$$

$$ParentWithBeautifulChild \equiv Parent \sqcap \exists hasChild.Pretty$$

$$Wife \equiv Woman \sqcap \exists hasHusband.Man$$

$$HealthyPerson \equiv Person \sqcap MentallyStable \sqcap EmotionallyStable \sqcap MedicallySound$$

$$\}$$

$$AB_F = \{$$

$$hasChild(MARY, PETER)$$

$$hasChild(PETER, HARRY)$$

$$hasChild(MARY, PAUL)$$

$$hasHusband(MARY, JOHN)$$

$$Female(MARY)$$

$$Person(MARY)$$

$$Person(PETER)$$

$$Person(HARRY)$$

$$Person(PAUL)$$

$$Person(JOHN)$$

$$Pretty(PETER, 0.6)$$

$$Pretty(MARY, 0.7)$$

$$MentallyStable(MARY, 0.1)$$

$$EmotionallyStable(MARY, 0.3)$$

$$MedicallySound(MARY, 0.4)$$

$$MentallyStable(PETER, 0.3)$$

$$EmotionallyStable(PETER, 0.1)$$

$$MedicallySound(PETER, 0.1)$$

$$MentallyStable(HARRY, 0.3)$$

$$EmotionallyStable(HARRY, 0.4)$$

$$MedicallySound(HARRY, 0.1)$$

$$MentallyStable(JOHN, 0.3)$$

$$EmotionallyStable(JOHN, 0.2)$$

$$MedicallySound(JOHN, 0.4)$$

}

The family ontology was gotten from Baader and Werner (2002) and shown in the appendix II. From the above FSSO, the set of instances is {Mary, Peter, Harry, Paul, John} and the set of terminal concepts is {Person, Female, Pretty}. The interpretations of the concepts are as follows:

$$Person^I = \{Mary, Peter, Harry, Paul, John\}$$

$$Female^I = \{Mary\}$$

$$Pretty^I = \{(Mary, 0.7), (Peter, 0.6)\}$$

Therefore,

$$\langle I, M\rangle = \{(Person, \{Mary, Peter, Harry, Paul, John\}),$$

$$\big(Female, \{Mary\}\big),$$

$$(Pretty, \{(Mary, 0.7), (Peter, 0.6)\})\}.$$

3.3. The FSSO Reasoning Technique

The proposed algorithm reasons over crisp and vague concepts in a domain of interest based on fuzzy soft set theory. The reasoner takes as input a Fuzzy Soft Set Ontology and a user query, to get the degree of truthness of the query passed, to make the reasoning, the reasoner goes through two major steps:

(1) Expansion step: this is where the query is expanded into its terminal concepts and the corresponding values asserted on the queried instance (or all instances if no instance passed with the query) over the terminal concepts in the AB_Fwill be retrieved.

(2) Construction of a comparison table with the values retrieved during the expansion as entries against each instances (as rows) and the terminal concepts (as columns). The r_i is the resultant fuzzy values (interpreted based on the semantics of the constructor involved) for each i, where $i = \{1, 2, \ldots, n\}$ are the instances in the domain.

Below is the FSSO reasoning algorithm:

Input:

A fuzzy soft set ontology $FSSO = \langle RB_F, TB_F, AB_F\rangle$

A query $Q \sqsubseteq FSSO$

Output: The degree of truthness of Q

Step 1: Generate an interpretation for the ontology-based fuzzy soft set $\langle I, M\rangle = \{(m_1, m_1^I), (m_2, m_2^I), \ldots, (m_n, m_n^I)\}$ in tabular form as follows:

a. Traverse through the TB_F and obtain terminal concept for all the non-terminal concept in Q.
b. Obtain the fuzzy value for each terminal concepts obtained in (a) above with respect to their corresponding instances:
    i. Value (1) if crisply asserted.
    ii. Value (0) if assertion not found
    iii. Fuzzy value for assertions with fuzzy value attached in their definition

Step 2:

If a valid instance in the domain is attached to Q, then:

a. Compute the resultant fuzzy value with respect to the relational operation(s) involved
b. Return the resultant fuzzy value as the degree of truthfulness for the instance parameter against the queried concept.

Else:

a. Construct the comparison table of the ontology-based fuzzy soft set $\langle I, M\rangle$ and compute the relational resultant for each instance $(r_i, \forall i)$
b. The optimal decision is to select $o_k\ if r_k = max_i\ \mathrm{r}_i$ and output o_k.
   If k has more than one value then any one of $r_k$may be chosen.

Algorithm 3. 1: Fuzzy soft set semantic making decision algorithm

3.4. Interpretation of Fuzzy Soft Set Ontology

The concept of fuzzy soft set is employed in interpreting vague concepts in a domain of discourse, by getting the fuzzy value of such vague concept as the satisfiable value or the value of 1 for the crisp concepts.

After getting all the values from each expanded concepts (1 and [0,1] for crisp and fuzzy concepts respectively), set operations union, intersection and complement are used to interpret the conjunction (⊓), disjunction (⊔) and negation (¬) constructors involved in the fuzzy soft set ontology.

3.4.1. FSSO Concept Constructors

Complex FSSO concepts are constructed from the terminal concepts in a domain with concept constructors in the same manner traditional DL constructors are used but the semantics are different in FSSO context.

The conjunction (known as intersection during interpretation) of any two or more concepts is the returned value common to all the concepts, since the smallest returned value of the concepts is also inclusive in all other participating concepts, consequently, the minimum returned value of all the concepts.

$$A \cap B \cap C \cap \ldots \cap P \ = \ min(A^I, B^I, C^I, \ldots P^I)$$

The disjunction (known as union during interpretation) of any two or more concepts is the returned value general to all the concepts, since the largest returned value of the concepts include the returned values of all other participating concepts, consequently, the maximum returned value of all the concepts.

$$A \cup B \cup C \cup \ldots \cup P \ = max(A^I, B^I, C^I, \ldots P^I)$$

The negation (known as complement during interpretation) of any concept is the returned value of the concept subtracted from 1.

$$\neg A \ = 1 - A^I$$

The existential relation *r* of a concept *A* to another concept *B* is the instance of *B* that has a role assertion with an instance of *A* with the least membership value.

The universal relation *R* of a concept *A* to another concept *B* is the instance of *B* with the greatest membership value that has a role assertion with an instance of *A*.

The at-least relation *r* of a concept *A* to another concept *B* require less than or exactly a specific number of instances of *A* to satisfy the role assertion $r(A^I, B^I)$.

The at-most relation r of a concept A to another concept B require greater than or exactly a specific number of instances of A to satisfy the role assertion $r(A^I, B^I)$.

Below is a summary of the syntax and semantics of FSSO concept constructors:

Table 3. 1: Syntax and Semantic of FSSO concept constructors

Constructor	**Syntax**	**Semantics**
Conjuction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Negation	$\neg C$	$1 - C^I$
Existential restriction	$\exists r.C$	$sup_{y \in \Delta^I}\{(x,y) \in r^I \wedge y \in C^I\}$
Value restriction	$\forall r.C$	$inf_{y \in \Delta^I}\{(x,y) \in r^I \longrightarrow y \in C^I\}$
At-least restriction	$\geq nr.C$	$\{x \in \Delta^I \mid \#\{y \in \Delta^I \mid (x,y) \in r^I \wedge y \in C^I\} \geq n\}$
At-most restriction	$\leq nr.C$	$\{x \in \Delta^I \mid \#\{y \in \Delta^I \mid (x,y) \in r^I \wedge y \in C^I\} \leq n\}$

Where inf (infimum) is greatest lower bound, sup (supremum) is least upper bound and # is the cardinality of a set.

Example 3.1: Interpretation of FSSO

Assuming the following concept definition: $HealthyPerson \equiv Person \sqcap MentallyStable \sqcap EmotionallyStable \sqcap MedicallySound$

Using the FSSO represented in section 3.2, the query parameter will give: Person, MentallyStable, EmotionallyStable and MedicallySound.

Table 3. 2: Comparison Table

i	$\boldsymbol{Person}$ a_1	$\boldsymbol{Mentally\ Stable}$ a_2	$\boldsymbol{Emotionally\ Stable}$ a_3	$\boldsymbol{Medically\ Sound}$ a_4	r_i
$\boldsymbol{MARY}$	1	0.1	0.3	0.4	0.1
$\boldsymbol{PETER}$	1	0.3	0.1	0.1	0.1
$\boldsymbol{HARRY}$	1	0.3	0.4	0.1	0.1
$\boldsymbol{PAUL}$	1	0	0	0	0
$\boldsymbol{JOHN}$	1	0.3	0.2	0.4	0.2

Table 3.1 has as column the obtained decision parameters {Person, MentallyStable, EmotionallyStable, MedicallySound} and as rows the instances in the FSSO domain {MARY, PETER, HARRY, PAUL, JOHN}. The individual entries e.g MARY = {1, 0.1, 0.3, 0.4} are the values gotten from the expansion of the corresponding decision parameters appearing on the column header. $\boldsymbol{r_i}$ is the resultant score values for each *i* obtained using the semantics of the corresponding constructor operator connecting the decision parameters. On Table 3.1 r_i evaluates to minimum of $\{a_1, a_2, a_3, a_4\}$ using the semantic of the constructor ⊓ that was used in connecting the decision parameters. Hence, the following is obtained $r_{\text{JOHN}} = 0.2, r_{\text{MARY}} = 0.1, r_{\text{PETER}} = 0.1, r_{\text{HARRY}} = 0.1, r_{\text{PAUL}} = 0$ from Table 3.1. So the decision is maximum r_i which is JOHN, therefore $HealthyPerson$ = JOHN (0.2).

Chapter Four

Implementation and Result

4.1. Implementing a FSSO Reasoner

A Fuzzy Soft Set Ontology was constructed by using a Poisson distribution to simulate its fuzzy values, the choice of Poisson distribution is based on its strength highlighted as follows:

(1) The event (instance) is something that can be counted in whole numbers;

(2) Occurrences are independent, so that one occurrence neither diminishes nor increases the chance of another;

(3) The rate at which events occur is constant. The rate cannot be higher in some intervals and lower in other intervals; and

(4) It is possible to count how many events have occurred.

The Poisson distribution is given as:

$$p(x\ numbers\ of\ fuzzy\ concept) = \frac{\lambda^x\ e^{-\lambda}}{x!}$$

Where:

- λ is the average number of events
- e is the exponential number (2.71828…)
- x is the discrete sample number (1,2,3,…, n) for each vague concepts in the ontology.

Below is the proposed algorithm for getting the fuzzy values of each vague concepts using the Poisson distribution.

Input:

n: number of instances

Output: f(i) fuzzy values in the range [0,1] for each i

i ⟵ 1;

Step 1: Get the average λ

for (y = 1 to n)

y ⟵ y + 1

λ ⟵ y / n

Step 2: Assign fuzzy values to each instance 1 <= i <= n

While n > i

$$f(i) \leftarrow \frac{\lambda^i e^{-\lambda}}{i!}$$

i ⟵ i + 1

Return $f(i)$

Algorithm 4. 1: Fuzzification

After constructing the FSSO, the content of the FSSO is saved in a text file (.txt extension), the file is then used in a program written in PHP language to demonstrate the implementation of the proposed FSSO reasoner. The working flow of the program is highlighted as follows:

(1) An interface to load the FSSO components (Role, Terminological and Assertional Boxes) as shown in Figure 4.1.

(2) After submitting the files in (1) above, an interface to allow users to make query into the FSSO domain is displayed as shown in Figure 4.2.

(3) Query must be written as any represented concept in the assertional box, the query will then be expanded to its terminal concepts, using the proposed FSSO reasoner algorithm, the degree of truthness of the queried concept will be returned as shown in Figure 4.3.

Figure 4. 1: File-load Interface

Figure 4. 2: Query Interface

Figure 4. 3: Query Output

4.2. Building the domain from FSSO

After the initial script that receives the DL ontology components (TBox and ABox files) executes by loading then with a simple interface as shown in Figure 4.1 above, the files are submitted to a script that implicitly parse all the concepts, roles and assertions of the ontology, the script parses each line on the ontology files. Each line in the TBox corresponds to a single concept definition while each line corresponds to a single assertion in the ABox. The parsing is based on the description on the syntactic diagram described in Figure 4.4 below.

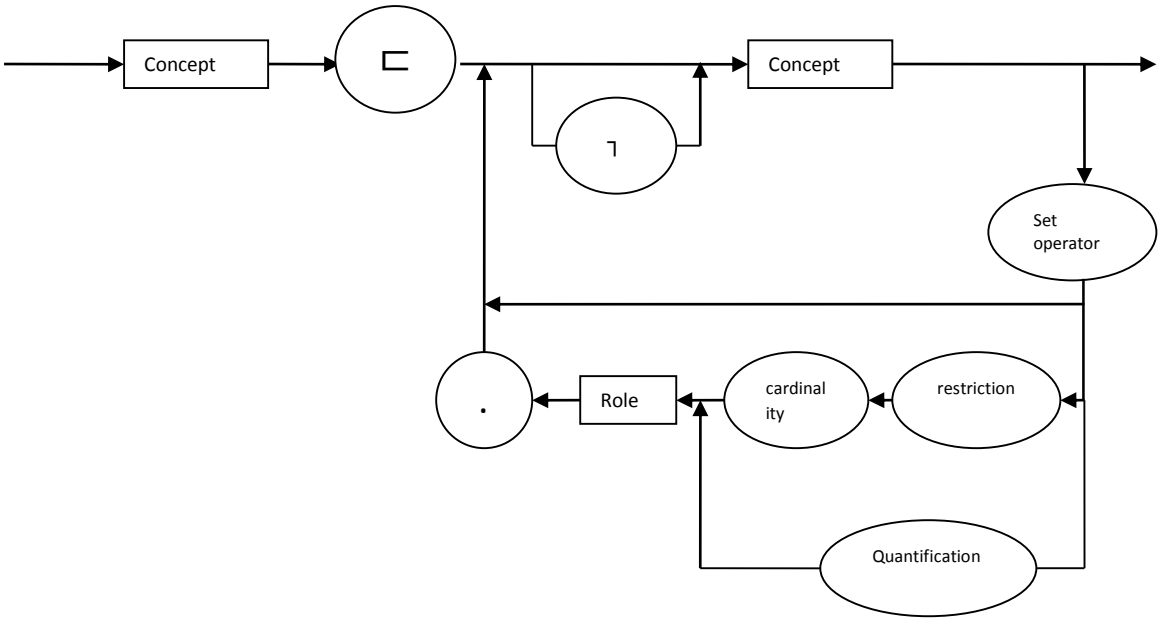

Figure 4. 4: DL General Expansion Diagram

The domain is then built up by making a call to a function (*expansion*) on each concepts in the TBox and/or RBox, the function explode all the concepts subsumed in the concept that is being expanded, then each concept encountered in the expansion is also expanded using the same *expansion* function until all the concepts are in their terminal stage. Then terminal concepts/relations are then traced into the ABox for any instance that is asserted with it.

In the process of doing the expansions, a tree like structure is formed where subsuming concepts could be viewed as a node with branches connecting to each of its component concepts until a terminal concept is reached which is then viewed as a node connecting each instances asserted to it in the ABox, those instances are the leafs in the tree structure. A graphical illustration of the expansion is shown in Figure 4.5 below.

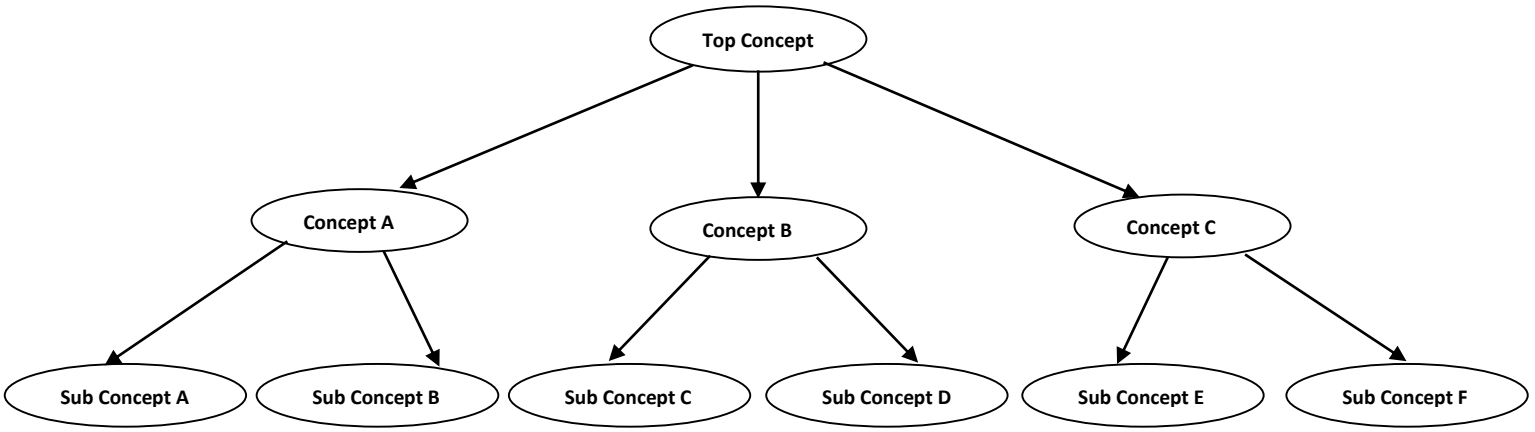

Figure 4. 5: Tree Structure of Concept Expansion

4.3. Making Queries into Domain

A query in the form of DL assertion can be received by a simple textbox, an assertion is expected to be in any of the following forms:

a. A terminology (concept) enclosing an individual (instance), which investigates whether the enclosed instance satisfies the terminology in the domain.
b. A terminology (relation) enclosing at least 2 individuals (instances), which investigates whether the enclosed instances are related through that terminology in the domain.
c. Just a terminology (concept or relation) without any instance enclosed, which investigates which of the existing instances in the domain satisfies that terminology the most, and/or what are the degree of satisfiablity of all the domain instances with respect to the terminology.

4.4. Getting Crisp or Fuzzy Value for Each Concept Expansion

From Section 4.2 above, where a tree like structure of the domain is made from the ontology constructs, a query of any form as discussed in Section 4.3 will undergo a set of transformational steps as follows:

a) Recursively replacing all concepts by their definitions, until only terminal concepts are in the formula (also called “unfolding” or “TBox-elimination”).
b) Transforming the main relation into a corresponding expression that allows for satisfiability testing. The resulting formula is called the goal.
c) Putting the expression into Negation Normal Form, where all negations move “inwards” in sub expressions, until only terminal concepts are negated.

For each leaf of the tree a value is being attached, which is either a one (1) or a fuzzy value [0, 1] denoting that the instance is crisply or fuzzily asserted respectively in the ABox

The pseudo codes for the construction of the tree together with the parsing of ontology file can be stated as follow:

```
While NOT End_Of_File (ontology file) Do
        variable_line = Readline(line)
        If (variable_line has component) Then
                term = Expand(variable_line)
        Else
                Attach_value(term)
End;
```

Algorithm 4. 2: Reading in ontology files

Both the Expand and Attach_value functions from the above pseudo code can also be stated as follows:

```
Expand(concept){
        For Each concept In variable_line
                If not_terminal(concept)
                        Expand(concept)
                Else
                        Return
        End;
}
```

Algorithm 4. 3: Expand pseudo code

```
Attach_value (term)
        If term is Fuzzy
                term_value = [0, 1]
        Else
                term_value = 1
        End;
}
```

Algorithm 4. 4: Attach_value pseudo code

4.5. Outputting Result

The reasoner will always output a result after a query is made into the domain regardless of the query containing a fuzzy concept or not. As such a reasoning algorithm based on degree of membership is proposed to carry out this task.

The fuzzy soft set membership degree is attached to any fuzzy concept in the ontology as represented by the ontology builder and the value 1 is attached to the crisp concepts.

A result is established by performing conjunction or disjunction operations on the resulting fuzzy membership value of the concepts involved in the query.

4.6. Comparison with Tableaux-Based Reasoning Procedure

The comparison between the fuzzy soft set reasoning technique presented in Section 3.4 and the tableaux based reasoning procedure discussed in Section 2.2.4.3 is based on the satisfiability of the example ontology used in Section 3.5. Tableaux procedure aims at constructing a model that satisfies all axioms of the given knowledge base. It is implemented as a finite tree which expands concepts using the expansions rule (Figure 2.3). Nodes of the tree are labelled with concept name and edges are labelled with role occurring between concepts. Like the algorithm presented in this work, the tableaux-based algorithm assumes the concepts' definition to be in NNF.

Example 4.1:

Satisfiability model of $ParentWithBeautifulChild$ using tableaux-based algorithm

$$ParentWithBeautifulChild \equiv Parent \sqcap \exists hasChild.Pretty$$

Step1: Expansion of $ParentWithBeautifulChild$based on the concept definition in TBox defined in Example 3.1.

$$ParentWithBeautifulChild \equiv \{Parent \sqcap \exists hasChild.Pretty\}$$

$$\equiv \{(Father \sqcup Mother) \sqcap \exists hasChild.Pretty\}$$

$$\equiv \{\big((Man \sqcap \exists hasChild.Person) \sqcup (Woman \sqcap \exists hasChild.Person)\big) \sqcap \exists hasChild.Pretty\}$$

$$\equiv \Big\{\Big(\big((Person \sqcap \neg Female) \sqcap \exists hasChild.Person\big) \sqcup \big((Person \sqcap Female) \sqcap \exists hasChild.Person\big)\Big) \sqcap \exists hasChild.Pretty\Big\}$$

Step 2: Construction of a tree model for concept $ParentWithBeautifulChild$ is illustrated in Figure 4.6.

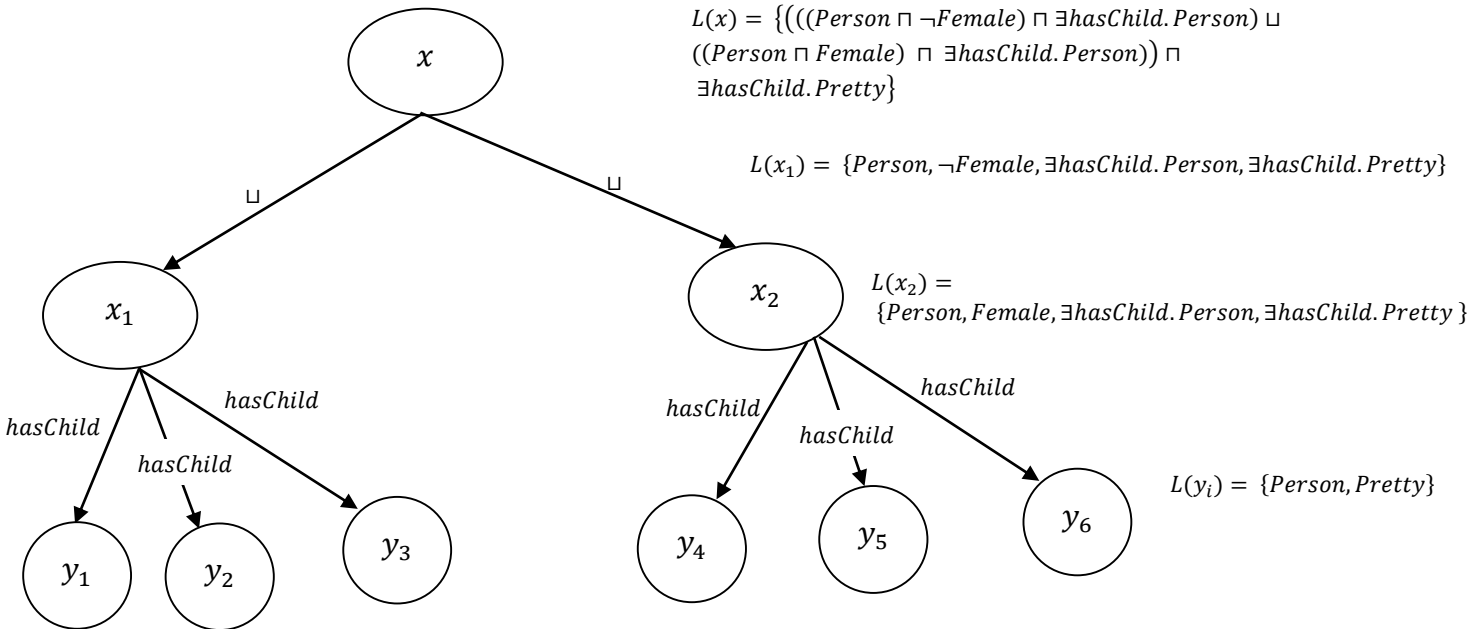

Figure 4. 6: Tree Model for ParentWithBeautifulChild

From knowledge represented in fuzzy soft set ontology in Example 3.1, the following can be established:

$Pretty^I = \{(PETER, 0.6)\}$

$hasChild^I(MARY) = \{PETER, PAUL\}$

From the tree model, the relation $hasChild(MARY)$ is satisfied. However, $\exists hasChild.Pretty$ is not satisfied since $\{PETER, PAUL\}$ are instances of the relation $hasChild(MARY)$ and the instance of $Pretty^I$ is fuzzy in nature which cannot be interpreted by the tableaux algorithm as an instance. This, violates the requirement of the universal quantification. Therefore, *ParentWithBeautifulChild* is not satisfiable with respect to the interpretation.

4.7. Comparison with Soft Set Based Ontology Reasoning Procedure

Again the comparison with the soft set reasoning procedure discussed in Section 2.2.4.4 is based on the satisfiability of the example ontology used in Section 3.5.

Example 4.2:

Satisfiability model of $ParentWithBeautifulChild$ using soft set semantic decision making algorithm

$$ParentWithBeautifulChild \equiv Parent \sqcap \exists hasChild.Pretty$$

According to the knowledge represented in Section 3.5, the domain here contain set of people, consequently, the instances in domain can be deduced as:

$$\Delta^I = \{MARY, PETER, HARRY, PAUL, JOHN\}$$

The decision parameters in the domain can also be deduced as:

$$M = \{Female, Person, Pretty, MentallyStable, EmotionallyStable, MedicallySound\}$$

Now, using the terminal expansion of the tableaux procedure in Section 4.6 above, the query parameters are:

$$Q = \{Person, Pretty\}$$

Let $\langle J, N \rangle$ be the ontology-based soft set from the query. The tabular representation of $\langle J, N \rangle$ is shown below:

Table 4. 1: Tabular representation of $\langle J, N \rangle$

i	$Person$	$Pretty$	$Choice\ Value$ c_i
$MARY$	1	0	0
$PETER$	1	0	0
$HARRY$	1	0	0
$PAUL$	1	0	0
$JOHN$	1	0	0

From Table 4.1, it is clear that all $c_i, i = \{MARY, PETER, HARRY, PAUL, JOHN\}$ evaluates to same value 0, making $max_i\ c_i = 0$, hence no instance in the domain are interpreted to be pretty.

Consequently, no instance in the domain with the relation $hasChild$ which are $hasChild^I = \{MARY, PETER\}$ will be considered $ParentWithBeautifulChild$. Hence, $ParentWithBeautifulChild^I$ will be an empty set.

This also violates the requirement of satisfiability since the instance $MARY$ has an instance of $Pretty^I$ (although fuzzy in nature) it is related with through the relation $hasChild$. Therefore, *ParentWithBeautifulChild* is not satisfiable with respect to the interpretation.

4.8. Fuzzy Soft Set Ontology Satisfiability Reasoning

Example 4.3:

Satisfiability model of $ParentWithBeautifulChild$ using the proposed fuzzy soft set ontology reasoning algorithm.

The Δ^I, M and Q remain as obtained in Section 4.7, the comparison table is shown below

Table 4. 2: Fuzzy Soft Set Comparison Table for $ParentWithBeautifulChild$

i	$Person$	$Pretty$	$Score\ Value$ r_i
$MARY$	1	0	0
$PETER$	1	0.6	0.6
$HARRY$	1	0	0
$PAUL$	1	0	0
$JOHN$	1	0	0

From Table 4.2, it could be seen that $PETER$ score value r_{PETER} evaluates to 0.6 making $Pretty^I = \{(PETER, 0.6)\}$. With the fuzzy soft set reasoner, $\exists hasChild.Pretty$ will evaluated to:

$hasChild^I(MARY) = \{(PETER, 0.6), (PAUL, 0)\}$

$hasChild^I(PETER) = \{(HARRY, 0)\}$

Therefore, $ParentWithBeautifulChild^I$ will return $MARY$ with the degree of truthiness as 0.6.

4.9. Fuzzy Soft Set Ontology Satisfiability with Crisp Ontology

It is necessary to also test the satisfiablity reasoning of the proposed algorithm if only crisp concepts are involved in an ontology representation or the query concept only has crisp concepts in its expansion. Below is an example of a query that only has crisp concepts in its expansion.

Example 4.4:

Satisfiability model of $MotherWithoutDaughter$ using the proposed fuzzy soft set ontology reasoning algorithm.

The expansion is as follows:

$$MotherWithoutDaughter \equiv Mother \sqcap \forall hasChild.\neg Female$$

$$\equiv (Woman \sqcap \exists hasChild.Person) \sqcap \forall hasChild.\neg Female$$

$$\equiv \big((Person \sqcap Female) \sqcap \exists hasChild.Person\big) \sqcap \forall hasChild.\neg Female$$

The comparison table:

Table 4. 3: Fuzzy Soft Set Comparison Table for $MotherWithoutDaughter$

i	$Person$	$Female$	$\exists hasChild.Perso$	$\forall hasChild.\neg Female$	$Score\ Value$ r_i
$MARY$	1	1	1	1	1
$PETER$	1	0	1	0	0
$HARRY$	1	0	0	0	0
$PAUL$	1	0	0	0	0
$JOHN$	1	0	0	0	0

From Table 4.3, it could be seen that $MARY$ score value r_{MARY} evaluates to 1 and all other instances have their score values to be 0, making $MotherWithoutDaughter^I = \{MARY\}$.

Therefore, $MotherWithoutDaughter^I$ will return $MARY$ with the degree of truthiness as 1.

4.10. Complexity Analysis

In this section, an analysis of the performance of the satisfiability algorithm with respect to its running time is discoursed.

Let *n* be the number of concepts of an ontology in consideration and *m* the number of terminal concepts. To expand a concept say *c*, the expansion will introduce at most $(n - m)$ new concepts to be expanded (since each concepts is expanded only once). Assuming each concept takes a constant time $O(1)$, the total time required for the expansion is:

$$T(n) = (n - m) \times O(1) = O(n).$$

Similarly, the retrieval of each expanded concept's value (fuzzy or crisp value) is linear since each assertion of a concept with an instance can only have a single value ([0-1], 1 or 0), hence the running time of *m* concepts value retrieval $T(m)$ is:

$$T(m) = O(m).$$

Finally, the worst case running time of the algorithm is (Time of expansion × Time of concepts instantiation)

$$T(n) = T(n) \times T(m)$$

$$= O(n) \times O(m)$$

$= O(n^2)$ if $n = m$.

Therefore running time of the FSSO satisfiability reasoner is of order n^2.

The running time of both Tableaux and Soft Set Ontologies is $O(n)$ since it both only deals with expansion and no retrieval of values. Therefore, there is an overhead of $O(n)$ in FSSO reasoner for its retrieval step.

Chapter Five

Summary, Conclusion and Recommendation

5.1. Summary

This dissertation addresses the limitation of Jiang *et al.* (2010) of not being able to reason over fuzzy concepts that cannot easily be specified as either true or false but better quantified in the form of degree, taking possible values of real number in the range of [0,1] and provides an efficient and flexible methods to interpret such vague concepts in an ontology in order to reason with the uncertain aspect of the world domain. This is necessary as a result of the inadequacy of traditional ontologies not been able to reason over vague concepts in knowledge-based of an application domain and also to perform inference with such vague knowledge caused by lack of having a fully and universally accepted understanding of a domain of discourse.

In a bid to carry out the study, Ontologies are viewed as a collection of well-structured set of object whereby DL is used to represent the knowledge about the collection. The approach of DL's RBox, TBox and ABox were extended to establish a Fuzzy Soft Set Ontology, $FSSO = \langle RB_F, TB_F, AB_F \rangle$, where RB_F, TB_F, and AB_F are the RBox, TBox, and ABox of FSSO respectively. From this conception of ontology the notion of fuzzy soft set is used to assign a fuzzy value to non-crispy concepts in the ontology domain so as to be able to reason with the uncertain part of the world. The fuzzy soft set method of comparison for getting the optimal decision is then used as the interpretation function for both the crisp and fuzzy concepts in the domain. The approach used in this study only extends the existing models in the field of ontology modelling with inclusion of fuzzy values to vague concepts and avoidance of remodelling the whole ontology.

5.2. Conclusion

The results obtained from this research work shows that fuzzy value can be attached to ontological concepts, which by extension could provide efficient and flexible methods to interpret uncertainty in an ontology in order to reason with the uncertain aspect of the world domain. Thus with the approach of this study, any query involving fuzzy concept made into the proposed fuzzy soft set ontology will always be satisfiable.

This dissertation was able to achieve the following contributions:

a) Extension of description logics with fuzzy soft set theory to resolve uncertainty in ontologies in order to be able to reason on the uncertain aspect of real world domain.
b) Development of a fuzzy soft set based satisfiability reasoning algorithm for fuzzy DL ontologies.

This research work can be helpful and applicable in real world problem such as data analysis in areas such as economics, engineering, environmental sciences, sociology, social sciences and medical sciences, data mining and forecasting.

5.3. Recommendation

The proposed approach is only based on DL ontologies with their RBox being empty. It is likely to have a real world scenario where uncertain roles are involved, as such future researches can tour the line of handling role uncertainty in an ontology. Furthermore, since there are many types of DL languages, it will be a worthwhile research effort to investigate into the likelihood of extending fuzziness to all types of DL representations.

To extend this work, one may possibly use ontology based fuzzy soft sets to address group decision making problems. An interesting topic of future research is to investigate semantic decision making using ontology-based fuzzy soft sets in a group decision making problems. It

is also desirable to further explore the applications of using the ontology-based fuzzy soft set approach to solve real world problems such as data mining, forecasting, and data analysis.

References

Atanassov, K., and Gargov, G. (1989). *Interval-valued intruitionistic fuzzy sets*. Fuzzy Sets and Systems, 31(4): 343-349.

Baader, F. and Werner, N. (2002). Basic Description Logics. In *The Description Logics Handbook*. F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider Eds. Cambridge: Cambridge University Press. pp. 43-95

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. (2007). *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd ed., Cambridge University Press.

Baader, F., Sattler, U. (2001). *An Overview of Tableaux Algorithms for Description Logics*. Studia Logica, 69(1): 5-40.

Bellenger, A. and Gatepaille, S. (2011). *Uncertainty in Ontologies: Dempster-Shafer Theory for Data Fusion Applications*. Computing Research Repository abs/1106.3876.

Berners-Lee, T., Hendlar, J. and Lassila, O. (2001). *The semantic web*. Scientific American, 279(1): 29-37.

Cagman, N., Enginoglu, S., and Citak, F. (2011). *Fuzzy Soft Set Theory and its Applications*. Iranian Journal of Fuzzy Systems, 2(1): 137-147.

Cheng, Y., Xiao, Z., and Lin, H. (2013). *A Fuzzy Soft Set Approach to Decision Making Problems Based on the Optimum Object for Ideal and Spatial Distance*. Advances in information Sciences and service Sciences(AISS), 965-974.

Costa, P. C., and Laskey, K. B. (2006). *PR-OWL: A Framework for Probabilistic Ontologies*. Frontiers in Artificial Intelligence and Applications, 150(1): 237-249.

Essaid, A., and Yaghlane, B. B. (2011). *BeliefOWL: An Evidential Representation in OWL Ontology*. International Semantic Web Conference. pp. 77-80.

Feng, F., Jun, Y. B., Liu, X., and Li, L. (2010). *An adjustable approach to fuzzy soft set based decision making*. Journal of Computational and Applied Mathematic, 5(1): 10-20.

Gruber, T. R. (1993). *A translation approach to portable ontologies*. Knowledge Acquisition, 5(2): 199-220.

Hunter, J. K. (2013). *An Introduction to Real Analysis*. https://www.math.ucdavis.edu/~hunter/intro_analysis_pdf/ch1.pdf. retrieved on 2nd February, 2016.

Horrocks, I. (1997). *Optimising Tableauxx Decision Procedures For Description Logics*. (Unpublished Ph.D thesis), University of Manchester, United Kingdom.

Horrocks, I. (2002). *Reasoning with expressive description logics: Theory and practice*. Proceeding of the 19th International Conference on Automated Deduction (CADE 2002), pp. 1-15.

Jiang, Y., Liu, H., Tang, Y., and Chen, Q. (2010). *Semantic decision making using ontology-based soft sets*. Mathematical and Computer Modelling, 42(11): 1005-1009.

Jiang, Y., Tang, Y., Chen, Q., Liu, H., and Tang, J. (2010). *Interval-valued intuitionistic fuzzy soft sets and their properties*. Computers and Mathematics with Applications, pp. 906-918.

Jiang, Y., Tang, Y., Chen, Q., Wang, J., and Tang, S. (2009). *Extending soft sets with description logics*. International Summer School. 160(23): 3403-3424.

Kana, A. F., and Akinkunmi, B. O. (2014). *An Algebra of Ontologies Approximation*. Journal of Intelligence Science, 4(2): 54-64.

Kana, A. F., and Akinkumi, B. O. (2016). *Modeling Uncertainty in Ontologies using Rough Set*. International Journal of Intelligent Systems and Applications. 8(4): 49-59

Keet, C. M. (2014). *Knowledge Engineering and Knowledge Management: Ontology Engineering*. Creative Commons. pp. 225-237.

Klinov, P. and Parsia B., (2008). *Pronto: a Non-Monotonic Proba-bilistic Description Logic Reasoner*. Procedings of the 5th European Semantic Web Conference, June 1-5, 2008. Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann Manolis Koubarakis Eds., Tenerife, Canary Islands, Spain. pp. 822-826.

Kong, Z., Gao, L., and Wang, L. (2009). *Comment on "A fuzzy soft set theoretic approach to decision*. Journal of Computational and Applied Mathematics. 223(2): 540-542.

Laskey, K. J., Laskey, K. B., Costa, P. C., Kokar, M. M., Martin, T. and Lukasiewicz, T. (2008). *Uncertainty Reasoning for the World Wide Web*. Report on the URW3-XG Incubator Group. http://www.w3.org/2005/Incubator/urw3/XGR-urw3/ retrieved 10 January, 2016.

Levy, Y. A., and Rousset, M. C. (1998). *Combining horn rules and description logics in carin*. Artificial Intelligence. pp. 5238-5245.

Maji, P. K., Biswas, R., and Roy, A. R. (2001). *Fuzzy soft sets*. Journal of Fuzzy Math, 9(3): 589-602.

Miller, G. A. (1995). *A Lexical Database for English*. Communication of the ACM. 100(1): 449-462.

Molodtsov, D. (1999). *Soft set theory - First results*. Computers and Mathematics with applications 37(4-5): 19-31.

Nardi, D. and Brachman, J. (2003). *An Introduction to Description Logics*. The Description Logic Handbook: Theory, Implementation, and Applications. F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider Eds. Cambridge University Press. pp. 43-95.

Pan, R., Ding, Z., Yu, Y., and Peng, Y. (2005). *A Bayesian Network Approach to Ontology Mapping*. Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland, November 6-10, 2005.

Pawlak, Z. (1982). *Rough sets*. International Journal of Information and Computer Sciences 11(5):341-356.

Richardson, R. (1994). *A Semantic-based Approach to Information Processing* (Unpublished Ph.D thesis). School of Computer Applications, Dublin City University, Ireland.

Roy, A. R., and Maji, P. K. (2007). *A fuzzy soft set theoretic approach to decision making problems*. Elsevier Journal of Computational and Applied Mathematics. 203(2): 412-418.

Sanchez, D. and Tettamanzi, A. (2006). *Reasoning and quantification in fuzzy description logics*. Capturing Intelligence: Fuzzy Logic and the Semantic Web. 1(1): 135-159.

Schmidt-Schauß, M., and Smolka, G. (1991). *Attribute concept descriptions with complements*. Journal of Artificial Intelligence. 48(1): 1-26.

Smith, B. (2003). *Blackwell guide to the philosophy of computing and information*. In Chapter Ontology, pp. 61-69.

Sowa, J. (2000). *Knowledge representation: logical, Philosophical and computational foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.

Stoilos, G., Stamou, G., Z., P. J., Tzouvaras, V. and Horrocks, I. (2007). *Reasoning with Very Expressive Fuzzy Description Logics*. Journal of Artificial Intelligence Research, 30(8): 273-320.

Straccia, U. (2001). *A Fuzzy Description Logic for the Semantic Web*. Artificial Intelligence Res., 137-166.

Uschold, M. and Gruninger, M. (1996). *Ontologies: principles, methods and applications*. Knowledge Engineering Review, 11(2): 93-136.

Zadeh, L. A. (1965). *Fuzzy Sets*. Information and Control 8(3): 338-353.

Zhao, J. (2010). *Uncertainty and Rule Extensions to Description Logics and Semantic Web Ontologies*. Advances in semantic computing. 2(1): 1-22.

Appendix I

Index.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="../references/assets/jquery-ui.css">
  </head>
  <body>
    <diV id="file_uploads">
      <form action="globals.php" method="post" enctype="multipart/form-data">
        <table>
          <tr>
            <td><label for="rfile">Role Box (RBox):</label></td><td><input type="file"
name="rfile" id="rfile"/></td>
          </tr>
          <tr>
            <td><label for="tfile">Terminological Box (TBox):</label></td><td><input type="file"
name="tfile" id="tfile"/></td>
          </tr>
          <tr>
            <td><label for="afile">Assertional Box (ABox):</label></td><td><input type="file"
name="afile" id="afile"/></td>
          </tr>
          <tr>
            <td colspan="2" align="right"><input type="submit" value="Submit"></td>
          </tr>
        </table>
      </form>
    </diV>
  </body>
</html>
```

Globals.php

```
<?php

session_start();

$file_abox = $_FILES["afile"]["tmp_name"];
$file_tbox = $_FILES["tfile"]["tmp_name"];

$operator = '';
```

```
$op = '';
$table1 = '';
$table2 = '';
$result = '';

$_SESSION['object'] = array();
$_SESSION['tbox_array'] = array(array());
$_SESSION['abox_array'] = array(array());
$symbols = array(
   "not" => "!",
   "and" => "&",
   "or" => "|",
   "for-all" => "#",
   "exist" => "*",
   "equal" => ":",
   "greater" => ">",
   "less" => "<",
   "greater-equal" => "/",
   "less-equal" => "\\"
);

if (file_exists($file_abox)) {
   $file_handle = fopen($file_abox, "r");
   while (!feof($file_handle)) {
      $line_of_text = fgets($file_handle);
      $box = explode(",", $line_of_text);
      if (count($box) == 1) {
         $first = trim($box[0]);
         $fbox = explode("(", $first);

         $_SESSION['object'][] = strtoupper(trim($fbox[1], ")"));

         $_SESSION['abox_array'][strtoupper(trim($fbox[1], ")"))][strtoupper($fbox[0])] = 1;
      } elseif (count($box) == 2) {
         $first = trim($box[0]);
         $second = trim($box[1]);
         $fbox = explode("(", $first);

         $_SESSION['object'][] = strtoupper($fbox[1]);

         if (!is_numeric(trim($second, ")"))) {

$_SESSION['abox_array'][strtoupper($fbox[1])][strtoupper($fbox[0])][strtoupper(trim($second, ")"))]
= 1;
         } else {
            $_SESSION['abox_array'][strtoupper($fbox[1])][strtoupper($fbox[0])] =
floatval(trim($second, ")"));
```

```
        }
      } elseif (count($box) == 3) {
        $first = trim($box[0]);
        $second = trim($box[1]);
        $third = trim($box[2]);
        $fbox = explode("(", $first);

        $_SESSION['object'][] = strtoupper($fbox[1]);


$_SESSION['abox_array'][strtoupper($fbox[1])][strtoupper($fbox[0])][strtoupper(trim($second, ")"))]
= floatval(trim($third, ")"));
      }
   }
   $_SESSION['object'] = array_values(array_unique($_SESSION['object']));
   unset($_SESSION['abox_array'][0]);
   fclose($file_handle);
} else {
   print "ABox file doesn't exist";
   exit;
}

if (file_exists($file_tbox)) {
   $file_handle = fopen($file_tbox, "r");
   while (!feof($file_handle)) {
      $line_of_text = fgets($file_handle);
      if (strpos($line_of_text, "^") === false) {
        $tbox = explode("=", $line_of_text);
        $x = trim($tbox[1]);
        if ((strpos($x, "&") === false) && (strpos($x, "|") === false)) {
          $temp = array();
          if ((strpos($x, ".") !== FALSE) && (preg_match('/[^0-9]/', $x) == 0)) {
            $temp = explode(".", $x);
            foreach ($temp as $key => $value) {
              $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['relation'][] = strtoupper($value);
            }
          } else {
            switch ($x) {
              case (strpos($x, ">") !== FALSE): {
                    $operator = ">";
                    $temp = explode($operator, $x);
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[0]);
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[1]);
                    continue;
                  }
              case (strpos($x, "<") !== FALSE): {
```

```
                $operator = "<";
                $temp = explode($operator, $x);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[0]);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[1]);
                continue;
            }
        case (strpos($x, ":") !== FALSE): {
                $operator = ":";
                $temp = explode($operator, $x);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[0]);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[1]);
                continue;
            }
        case ( strpos($x, "\\") !== FALSE): {
                $operator = "\\";
                $temp = explode($operator, $x);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[0]);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[1]);
                continue;
            }
        case (strpos($x, "/") !== FALSE): {
                $operator = "/";
                $temp = explode($operator, $x);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[0]);
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($temp[1]);
                continue;
            }
        default : {
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($x);
                continue;
            }
      }
    }
  } else {
    $tmp = array();
    $temp = array();
    switch ($x) {
      case (strpos($x, "&") !== false): {
            $tmp = explode("&", $x);
            $op = "&";
            continue;
        }
```

```
            case (strpos($x, "|") !== false): {
                $tmp = explode("|", $x);
                $op = "|";
                continue;
              }
            default :
              continue;
          }
          foreach ($tmp as $value) {
            if (strpos($value, ".") !== FALSE) {
              $temp = explode(".", $value);
              foreach ($temp as $key => $value) {
                $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['relation'][] =
strtoupper($value);
              }
            } else {
              switch ($value) {
                case (strpos($value, ">") !== FALSE): {
                    $operator = ">";
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper(trim($value,
$operator));
                    continue;
                  }
                case (strpos($value, "<") !== FALSE): {
                    $operator = "<";
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper(trim($value,
$operator));
                    continue;
                  }
                case (strpos($value, ":") !== FALSE): {
                    $operator = ":";
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                    $_SESSION['tbox_array'] [strtoupper(trim($tbox[0]))][] = strtoupper(trim($value,
$operator));
                    continue;
                  }
                case ( strpos($value, "\\") !== FALSE): {
                    $operator = "\\";
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper(trim($value,
$operator));
                    continue;
                  }
                case (strpos($value, "/") !== FALSE): {
                    $operator = "/";
```

```
                        $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['operator'] = $operator;
                        $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper(trim($value,
$operator));
                        continue;
                    }
                default : {
                    $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))][] = strtoupper($value);
                    continue;
                }
            }
        }
        }
        $_SESSION['tbox_array'][strtoupper(trim($tbox[0]))]['op'] = $op;
      }
    } else {
      $tbox = explode("^", $line_of_text);
      $_SESSION['tbox_array']['TopConcept'][] = strtoupper(trim($tbox[1]));
    }
  }
  unset($_SESSION['tbox_array'][0]);
} else {
  print "TBox file doesn't exist";
  exit;
}

$table1 .= "<table border='1'>";
foreach ($_SESSION['abox_array'] as $key => $value) {
  $table1 .= "<tr><td>" . $key . "</td>";
  foreach ($value as $key => $value) {
    if (is_array($value)) {
      $table1 .= "<td>" . $key . "</td>";
      foreach ($value as $k => $val) {
        $table1 .= "<td>" . $k . "</td>" . "<td>" . $val . "</td>";
      }
    } else {
      $table1 .= "<td>" . $key . "</td>" . "<td>" . $value . "</td>";
    }
  }
  $table1 .= "</tr>";
}
$table1 .= "</table>";

$table2 .= "<table border='1'>";
foreach ($_SESSION['tbox_array'] as $key => $value) {
  $table2 .= "<tr><td>" . $key . "</td>";
  foreach ($value as $key => $value) {
    if (is_array($value)) {
```

```
        foreach ($value as $k => $val) {
            $table2 .= "<td>" . $val . "</td>";
        }
    } else {
        $table2 .= "<td>" . $value . "</td>";
    }
  }
  $table2 .= "</tr>";
}
$table2 .= "</table>";
fclose($file_handle);
header("location:start.php");
?>
```

Start.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="../references/assets/jquery-ui.css">
  </head>
  <body>
    <?php
    session_start();
    // put your code here
    include_once './functions.php';
    ?>
    <diV id="selection">
      <form id="main_form" name="main_form" method="post" action="javascript:void(0)">
        <label for="query">Make query into the domain:</label>
        <input type="text" name="query" id="query" />
        <input type="submit" name="btn_query" id="btn_query" value="Query" />
      </form>
      <div style="width:100%" id="param_div"></div>
      <div style="width:100%" id="table_div"></div>
      <br />
      <div style="width:100%" id="pre_resultant"></div>
      <div style="width:100%" id="resultant_div"></div>

    </diV>

    <script type="text/javascript" src="./references/js/jquery.js"></script>
    <script type="text/javascript" src="./references/js/jquery-ui-1.8.17.js"></script>
```

```
<script type='text/javascript'>
    $("#btn_query").live("click", function () {
        var query = $("#query").val();
        $.ajax({
            url: "fuzzy_tables.php",
            type: 'POST',
            data: ({
                query: query
            }),
        }).done(function (msg) {
            $("#param_div").html(msg);
        });
        return false;
    });

    $("#btn_generate").live("click", function () {
        $.ajax({
            url: "fuzzy_tables.php",
            type: 'POST',
            data: $("#frm_dynamic_param").serialize()
        }).done(function (msg) {
            $("#table_div").html(msg);
            $("#operation").show();
        });
        return false;
    });

    $("#fuzzy_param").live("click", function () {
        var f_param = $("#fuzzy_param").val();
        $.ajax({
            url: "pre_resultant.php",
            type: 'POST',
            data: ({
                f_param: f_param
            })
        }).done(function (msg) {
            $("#pre_resultant").html(msg);
        });
        return false;
    });

    $("#btn_operation").live("click", function () {
        $.ajax({
            url: "resultant_table.php",
            type: 'POST',
            data: $("#frm_param_op").serialize()
        }).done(function (msg) {
```

```
            $("#resultant_div").html(msg);
          });
          return false;
        });
    </script>
  </body>

</html>
```

Functions.php

```
<?php

function permutations(array $array, $inb = false) {
  switch (count($array)) {
    case 1:
      return $array[0];
      continue;
    case 0:
      throw new InvalidArgumentException('Requires at least one array');
      continue;
  }
  $keys = array_keys($array);
  $a = array_shift($array);
  $k = array_shift($keys); // Get the key that $a hadreturn $a;
  $b = permutations($array, 'recursing');

  $return = array();

  foreach ($a as $k1 => $v) {
    foreach ($b as $k2 => $v2) {
      if ($inb == 'recursing')
        $return[] = array_merge(array($v), (array) $v2);
      else
        $return[] = array($k => $v) + array_combine($keys, $v2);
    }
  }

  return $return;
}

$_SESSION['return'] = array();
$_SESSION['concepts'] = array();

function expansion($var, $obj, $int = "", $restriction = "", $any = FALSE) {
  if (array_key_exists(trim($var, "!"), $_SESSION['tbox_array'])) {
```

```
      foreach ($_SESSION['tbox_array'][trim($var, "!")] as $key => $value) {
         if (is_numeric($key)) {
            if (preg_match('/^[0-9]/', $value) == 1) {
               $_SESSION['return'][] = expansion($value, $obj, "", substr($value, 0, 1));
            } else {
               if (strpos($var, "!") !== FALSE) {
                  $_SESSION['return'][] = 1 - expansion($value, $obj);
               } else {
                  $_SESSION['return'][] = expansion($value, $obj);
               }
            }
         } elseif ($key == "relation") {
            switch ($value) {
               case (strpos($value[0], "#") !== FALSE): {
                     $complement = (strpos($value[1], "!") !== FALSE) ? TRUE : FALSE;
                     $temp_any = expansion($value[1], $obj, "", "", TRUE);
//                     var_dump($temp_any);
                     $_SESSION['return'][] = search_any(trim($value[0], "#"), $obj, $temp_any,
$complement);
                     break;
                  }
               case (strpos($value[0], "*") !== FALSE): {
                     $complement = (strpos($value[1], "!") !== FALSE) ? TRUE : FALSE;
                     $temp_any = expansion($value[1], $obj, "", "", TRUE);
                     $_SESSION['return'][] = search_all(trim($value[0], "*"), $obj, $temp_any,
$complement);
                     continue;
                  }
               default : {
                     continue;
                  }
            }
         } elseif ($key == "op" || $key == "operator") {
            $_SESSION['return'][] = $value;
         }
      }
   } else {
      if ($any) {
         foreach ($_SESSION['abox_array'] as $key => $value) {
            $array[$key] = floatval($value[trim($var, "!")]);
         }
         $any = FALSE;
         return $array;
      } elseif (array_key_exists(trim($var, "!"), $_SESSION['abox_array'][$obj]) &&
!is_array($_SESSION['abox_array'][$obj][trim($var, "!")])) {
         if (strpos($var, "!") !== FALSE) {
            return floatval(0);
```

```
        } else {
            return floatval($_SESSION['abox_array'][$obj][trim($var, "!")]);
        }
    }else{
        if(strpos($var, "!") !== FALSE){
            return 1;
        }else{
            return 0;
        }

    }
  }
}

function search_any($relation, $obj, $array, $complement) {
  $var_r = 0;
  foreach ($array as $key => $value) {
    if ($value > 0 && array_key_exists($key, $_SESSION['abox_array'][$obj][$relation]) && $key !=
$obj) {
        $var_r = $value;
        break;
    }
  }
  if ($complement) {
    return 1 - $var_r;
  } else {
    return $var_r;
  }
}

function search_all($relation, $obj, $array, $complement) {
  $var_r = array();
  foreach ($array as $key => $value) {
    if ($value > 0 && array_key_exists($key, $_SESSION['abox_array'][$obj][$relation]) && $key !=
$obj) {
        $var_r[] = $value;
    } else {
        $var_r[] = 0;
        break;
    }
  }
  if ($complement) {
    return 1 - min($var_r);
  } else {
    return min($var_r);
  }
} ?>
```

Appendix II

Family TBox

Source: Baader and Werner (2002)

$$TBox = \{$$

$$Woman \equiv Person \sqcap Female$$

$$Man \equiv Person \sqcap \neg Female$$

$$Mother \equiv Woman \sqcap \exists hasChild.Person$$

$$Father \equiv Man \sqcap \exists hasChild.Person$$

$$Parent \equiv Father \sqcup Mother$$

$$Grandmother \equiv Mother \sqcap \exists hasChild.Parent$$

$$MotherWithManyChildren \equiv Mother \sqcap \geq 3hasChild$$

$$MotherWithoutDaughter \equiv Mother \sqcap \forall hasChild.\neg Female$$

$$ParentWithBeautifulChild \equiv Parent \sqcap \exists hasChild.Pretty$$

$$Wife \equiv Woman \sqcap \exists hasHusband.Man$$

$$\}$$