Fuzzy Ontology Merging for the Semantic Web

A dissertation submitted in partial fulfillment of the requirement for the award of the degree of

Master of Technology
In
Computer Science and Technology

By
L. Rajya Lakshmi

Under the supervision of
Prof. K. K. Bharadwaj

School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi – 110067
July 2008

जवाहरलाल नेहरू विश्वविद्यालय

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI - 110067 (INDIA)

Certificate

This is to certify that the dissertation "**Fuzzy Ontology Merging for the Semantic Web**" submitted by **Ms. L. Rajya Lakshmi** for the award of the degree of **Master of Technology** in **Computer Science and Technology** to the **Jawaharlal Nehru University** is a bona fide record of work carried out under the guidance and supervision of **Prof. K. K. Bharadwaj**. The results embodied in the dissertation have not been submitted to any other University or Institute for the award of any Degree or Diploma.

L. Rajya Lakshmi

L. Rajya Lakshmi

(Student)

[illegible] 18-07-08

Prof. K. K. Bharadwaj

(Supervisor)

[illegible] 24/7/08

Prof. Parimala N.

Dean SC&SS

Jawaharlal Nehru University

New Delhi – 110067 (INDIA)

Acknowledgments

I express my deep sense of gratitude to my supervisor **Prof. K. K. Bharadwaj** for his constant support, encouragement and constructive suggestions throughout my thesis work. His wide knowledge and logical way of thinking have been of great value for this thesis. I am grateful for his detailed and constructive comments.

I would like to acknowledge the support of **School of Computer and Systems Sciences** at the **Jawaharlal Nehru University** for providing me with resources to make this work possible. I am grateful to all faculty members for their support and cooperation in particular, **Prof. S. Balasundaram** and **Prof. Parimala N**.

I would also like to thank all my friends for their cooperation and encouragement. I thank my husband **Dr. S. Chandra Sekhara Rao** for his support and encouragement. Of course none of this would have been possible without the support of my children **Bhavyesh** and **Bhaagyesh**, my parents and in-laws.

Last but not the least I thank God for his guidance and blessings, and making all of this possible.

L. Rajya Lakshmi

Abstract

Ontologies are back bone of the semantic web. Ontology organizes domain knowledge in terms of concepts, properties and relations. It defines a common vocabulary for researchers who need to share information in a domain. But the most information related to the real world knowledge is ill-structured, uncertain and imprecise. The present structure of the ontology can not represent this information. A fuzzy ontology structure has been defined as an extension of the standard ontology structure to handle this information. Associations between properties and concepts and relations between concepts have been assigned membership values in the fuzzy ontology structure.

Many methods, tools, and techniques have been developed for crisp ontology merging. PROMPT, FCA – Merge, Chimaera, and ONION are examples of some of them. In this work we propose two fuzzy ontology merging algorithms. In the first algorithm called Similarity Based Fuzzy Ontology Merging (SBFOM) Algorithm, we combine linguistic similarity, lexical similarity, and contextual similarity to find similar concepts. The performance of SBFOM algorithm is evaluated by building local consensus fuzzy ontology. In the second algorithm called Fuzzy FCA-Merge (FFCA-Merge) algorithm, we extend FCA – Merge to fuzzy ontology merging and its performance is evaluated on synthetic data.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. 1 Introduction

The World Wide Web is a collection of electronic documents linked together like a spider web. These documents are stored on servers located around the world. A user can view a Web page that may contain text, image, video, and other multimedia data and can navigate between Web pages using hyperlinks among them with the help of a Web browser. As a result of this enormous usage, the web has evolved into a global electronic publishing medium and medium for conducting electronic commerce. This Web was mainly designed for exchange of documents.

Web contains virtually boundless information in the form of documents. By using computers we can search for documents. And computers can present us with the results, but cannot understand which result is the most relevant in a given circumstance. User has to go through the results to obtain the required information. But the amount of the data is so huge that only computers can process that data. Another problem with the results generated by search engines is that not all retrieved documents answer a user's query. In most cases precision is low. The assistance provided by directories (such as Yahoo!) and search engines (such as Google and AltaVista) are of only little useful.

Document retrieval is one of the applications of the Web. Users often want to use the Web for other purposes such as air ticket reservation, to make Hotel bookings, to find best price for some electronic good (such as a desktop computer) and many more. Completion of these tasks often involves visiting a series of pages, integrating their content and reasoning in some way. The capabilities provided by the present directories and search engines can not perform these tasks. This is because the information provided in the Web page cannot help computers in understanding the meaning of the text in a Web page.

Semantic Web has been defined as a solution to this problem. The vision of the Semantic Web is to add semantics to present Web documents to make the meaning of web pages explicit and convert them into data to be shared effectively by wider communities, and to be processed automatically by tools (computers) as well as manually.

1. 2 The Semantic Web

Tim Berners-Lee [Berners-Lee, et al, 2001], inventor of the Web, proposed the term Semantic Web, and defined the Semantic Web as follows:

"The Semantic Web is not a separate Web but extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. This is a web of data that can be processed by machines directly and indirectly".

Implementing the Semantic Web requires adding semantic metadata (data that describes data), to information sources. This will allow machines to process effectively the data based on the semantic information that describes it. When there is enough semantic information associated with the data, computers can make inferences about data, that is, they understand what a data resource is and how it is related to other data.

1. 2. 1 Applications of the Semantic Web

Improved search is one of many potential benefits from the Semantic Web. To provide better search results, the Semantic Web augments current web pages with markup that captures some of the meaning of the content on the web pages and encodes it in a form that is suitable for machine understanding and enables semantically empowered search engines [Shah, et al, 2002].

Internet agents, which are autonomous programs that interact with the Internet, can also benefit from the Semantic Web. In order to accomplish some goal an Internet agent, can request and perceive Web pages and execute Web services. Such agents are capable of comparison shopping, participating in an auction, or arranging a complete vacation. The existing agents to perform these tasks are built to handle only a predefined set of Web pages and are highly dependent on the structure of these pages. If a web page changes, the agent may no longer be able to locate information or interact with it. Agents that could consider the semantics of a web page instead of its layout would be much more robust.

Push systems are another area that could benefit from the Semantic Web. Push systems, instead of forcing the users to retrieve relevant information, the web pages are forced to the users. Such systems require a profile of a user and a method to evaluate whether a particular web page suits a user profile or not. If the web pages are annotated with the semantic information the evaluation method can provide more accurate results and can prevent unwanted pushing of the information to the user [Heflin and Hendler, 2001].

1. 2. 2 Layered Architecture of the Semantic Web

Berners-Lee has suggested a layered architecture for the Semantic Web to add semantic information. This architecture is shown in the Figure 1.1. According [Stumme, et al, 2006], on first two layers a common syntax is provided. Uniform resource identifiers (URIs) provide a standard way to refer to entities, while

Unicode is a standard for exchanging symbols. The Extensible Markup Language (XML) fixes a notation for describing labeled trees, and XML Schema allows the definition of grammars for valid XML documents. The Resource Description Framework (RDF) can be seen as the first layer where information becomes machine-understandable. According to W3C recommendation, RDF "is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web". RDF Schema defines a simple modeling language on top of RDF which includes classes, "is_a" relationships between classes and between properties, and domain/range restrictions for properties.

The next layer in this architecture is Ontology Vocabulary. According [Gruber, 1993], an ontology "is an explicit formalization of a shared understanding of a conceptualization". A detailed discussion on ontologies follows in next sections. The next layer is Logic. This is nothing but deducing new knowledge from the available information using logical reasoning. Next two layers are Proof and Trust. They follow the understanding that it is important to be able to

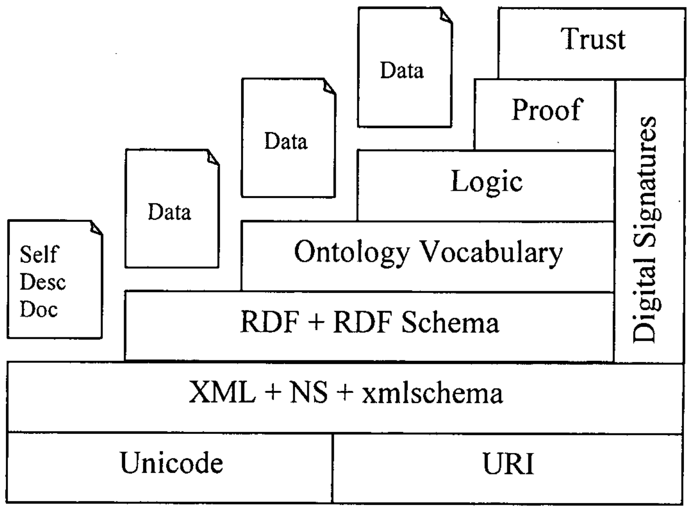


Figure 1.1. Berners-Lee's Layered Architectures of the Semantic Web

check the validity of statements made in the Semantic Web, and that trust in the Semantic Web and the way it processes information will increase in the presence of statements thus validated. So the author must provide a proof which should be verifiable by a machine.

Ontology is a representation of a set various types of objects and relationships among those objects. Ontologies are used in artificial intelligence, the Semantic Web, software engineering, biomedical informatics, and information architecture as a form of knowledge representation about the world or some part of it. Ontologies are the backbone of the Semantic Web.

1. 3 Ontologies

In this section we discuss, the uses of ontologies through examples. These examples also explain the meaning of ontology. Suppose a person wants to have information on "Jewellery". Now his/her search agent searches for the keyword "Jewellery". But some of the sites might have listed jewellery as "Ornaments". So the software agent searching for the keyword "Jewellery" must be able to know that "Ornaments" is a synonym of "Jewellery". And also the search agent must be able to reason that "Gold Jewellery" is a "kind of" "Jewellery". In order to provide this kind of knowledge for the search agent ontology is required.

Previous example explains the importance of structural organization of terms and relationships among them. Now we look at another example that explains the importance of context of terms. A word "Cricket" in the context of sports is different from the word "Cricket" in the context of insects. A person who wants to read articles on "Cricket" (insect) searches for "articles on Cricket". The search engine which is unaware of the context of the word "Cricket" will generate search results consisting of articles on "Cricket" in the context of sports and articles on "Cricket" in the context of insects. In order to obtain correct search results user has to provide the context of the search term.

Ontologies are key component of the Semantic Web. They facilitate a machine processable representation of information. They bridge the effective communication gap between users and machines.

1. 3. 1 Definitions of Ontology

Various definitions have been given for ontology in the literature. According to [Gruber and Olsen, 1994], formal ontologies serve as "specifications of common conceptualizations" among agents. Another definition given by Guarino in [Guarino, 1995] defines ontology as "A logical theory which gives an explicit, partial account of a conceptualization, designed in order to be shared by many agents for various purposes".

[Noy and Musen, 1999] defines ontology as a formal explicit description of concepts in a domain discourse (**classes** are sometimes called as concepts), properties of each concept describing various features and attributes of the concept (**slots** sometimes called roles or properties), and restrictions on slots (**facets** sometimes called restrictions).

There are five main relations existing between concept pairs. They are

- Is-Super-Class
- Is-Sub-Class
- Is-Part-of
- Contains
- Equivalent to

"C1 Is-Super-Class C2" implies C1 is generalization of C2, and C2 inherits all property descriptors of C1. For example in the Figure 1.2 "Animal Is-Super-Class Person" means "Person" inherits all properties of "Animal". "C2 Is-Sub-Class C1" implies C2 is specialization of C1, and C2 inherits all properties of C1. It is reverse of "Is-Super-Class". In the Figure 1.2 "Creature Is-Sub-Class Animate-

Being" means "Creature" inherits all properties of "Animate-Being". These two relation are represented by an arrow from C2 to C1 labeled with "Is-a" or "Kind-of". Relation "C1 Contains C2" is inverse of "C2 Is-Part-Of C1". Here C1 is defined as aggregation of other concepts.

"C1 Equivalent to C2" implies C1 and C2 are synonyms. For example "Person" and "Human" are equivalent and, "Living-thing" and "Animate-Being" are equivalent. This relation is represented by a double arrowed line labeled with "same-as" or "Eq". If two concepts C1 and C2 are equivalent then, C1's instances are inferred as C2's instances.

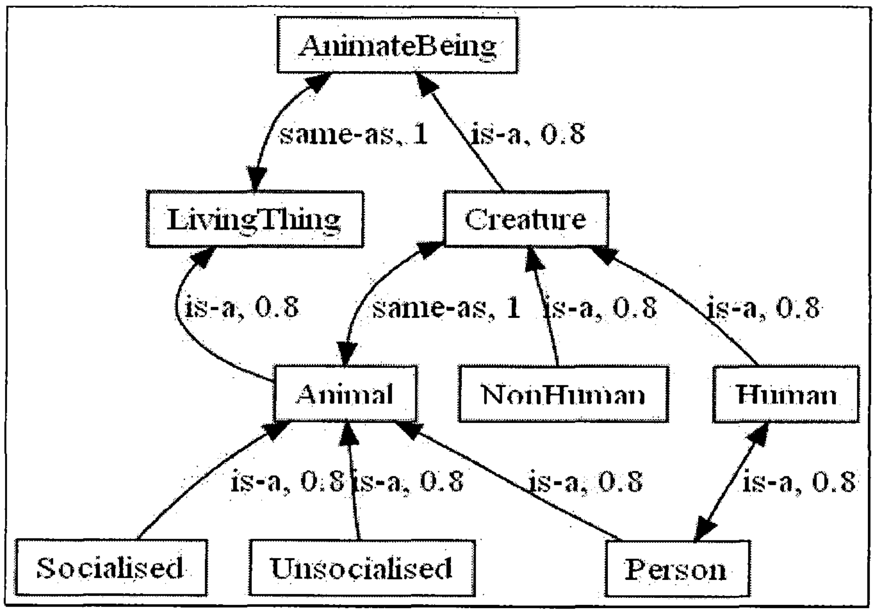


Figure 1.2. Fuzzy Ontology to explain various Relations between Concepts

1. 3. 2 Need for Ontology

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. Below we have mentioned some of the reasons behind the development of the ontologies.

1. To share common understanding of the structure of information among people or software agents. If a set of Web sites share and publish same underlying ontology for a particular domain, then the computer agents can extract and aggregate information from these different sites more efficiently.
2. To enable reuse of the domain knowledge.
3. To make domain assumptions explicit.
4. To separate domain knowledge from the operational knowledge. This is another common use of ontologies. Ontology can be used for the task of configuring a product from its components according to a required specification, and implement a program that does this configuration independent of the products and components themselves.
5. To analyze domain knowledge.

1. 4 Fuzzy Ontologies

The construction of ontology implies the parallel construction of a vocabulary for it. Gruber in [Gruber, 1993] pointed out that, "pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents." But most of the information which related to world knowledge is ill-structured, uncertain and imprecise. Usually in the Semantic Web, knowledge is assumed to be crisply defined and no uncertainty or imprecision is allowed in the description of objects. The Semantic Web deals with hard semantics in the description and manipulation of crisp data. For example they can handle sentences like "The Ganga is a river". It cannot represent sentences like "The Ganga is a *very_long* river" with soft semantics. This type of information can be processed by using fuzzy logic concepts and techniques [Zadeh, 1965]. "The Ganga is a *very_long* river" can be translated into "length (Ganga) is *very_long*". Here the term "very_long", is assumed as the label of a fuzzy set.

A fuzzy ontology structure can be defined as consisting of concepts, of fuzzy relations among concepts, of a concept hierarchy or taxonomy, of non-hierarchical associative relationships and of a set of ontology axioms, expressed in an appropriate logical language [Sanchez and Yamanoi, 2007]. Every fuzzy relation between two concepts associated with a membership value which corresponds to a fuzzy membership relation such as "strongly", "partially", "somewhat", and "slightly".

1. 4. 1 Fuzzy Ontology Definitions

In this subsection fuzzy ontology definitions provided by various authors are presented. Some of the authors have given methods for converting crisp ontologies into fuzzy ontologies.

David Parry in [Parry, 2004b] modified the crisp ontology structure to fuzzy ontology structure. That modification is entirely incremental, conversion to fuzzy ontology adds membership values to currently existing relations, and may also add new entries, in the ontology. The ontology membership is normalized in respect to each term in the ontology that is the sum of the membership values of each term in the ontology is equal to one. He has used the fuzzy ontology for the mapping between query terms and members of the ontology, because the relative importance of a particular mapping to an overloaded term may be different for different user, and this information is very important for accurate satisfaction of a query.

In [Abulaish and Dey, 2006], a fuzzy ontology structure is created as an extension to the standard ontology structure. In their fuzzy ontology structure a concept descriptor is represented as a fuzzy relation, which encodes the degree of a property value using a fuzzy membership function. They stored the concept descriptions in a <property, value, qualifier, constraints> framework, where the value and qualifier both defined as fuzzy sets. This framework allows for defining the property value of a concept with varying degree of fuzziness without changing the concept description paradigm. Such concept descriptions termed as imprecise. Other than concept descriptors, other relations in the ontology Is-a and Has-part etc. are also associated with a strength of association.

According to [Tho, et al, 2006], a fuzzy ontology $\boldsymbol{F_o}$ consists of four elements ($\boldsymbol{C}$, $\boldsymbol{A^C}$, $\boldsymbol{R}$, $\boldsymbol{X}$), where $\boldsymbol{C}$ represents a set of concepts, $\boldsymbol{A^C}$ represents a collection of attributes sets, one for each concept, and $\boldsymbol{R} = (\boldsymbol{R_T}, \boldsymbol{R_N})$ represents a set of relationships, which consists of two elements: $\boldsymbol{R_N}$ is a set of nontaxonomy relationships and $\boldsymbol{R_T}$ is a set of taxonomy relationships. $\boldsymbol{X}$ is a set of axioms. Each axiom in $\boldsymbol{X}$ is a constraint on the concept's and relationship's attribute values or a constraint on the relationships between concept objects.

1. 4. 2 Advantages of Fuzzy Ontologies

The most important application area of fuzzy ontologies is search process. The use of fuzzy ontology for mapping the search terms allows the relative weight of each term in the required output to be calculated. By allowing these weights to be calculated accurately, it removes the bias associated with multiply located terms being used for searching.

The use of fuzzy ontology approach allows the convenient representation of the relationships in a domain according to a particular view, without sacrificing commonality with other views. The ontology framework is common, just the membership values are different.

Many issues arise from the use of multiple ontologies, including the difficulties associated with communicating between ontologies and the need for maintenance of large number of ontologies. The fuzzy ontology can be used to define a common frame work, or a base ontology, with different membership values associated with different users and groups.

Finally this approach holds out the possibility that the representation of a potentially very large ontology can be compressed. If whole areas are not required, the relations to the core can be set to zero. Unwanted intermediate levels can also be removed, with lower level terms only communicating directly with higher level terms. This aspect removes the need to create artificial groupings to avoid orphaned terms [Parry, 2004a].

In [Widyantoro and Yen, 2001], authors have used a fuzzy ontology structure for query refinement. They have find out Narrower-term and Border-term relations between pairs of terms to find the ontology structure. After building the fuzzy ontology in this way they have used that ontology to find Narrower-terms set and Border-terms set for a given query term. Then they have replaced query term with a term (query refinement term) from the either set or used the query term conjunctively with the selected term to refine query. In this way the query refinement term helps in shifting the search focus to a more specific context.

In [Lee, et al, 2005] a fuzzy ontology is used for news summarization. A fuzzy ontology can also be used to improve semantic documents retrieval [Calgeri and Sanchez, 2007].

In this thesis we developed two fuzzy ontology merging algorithms: Similarity based fuzzy ontology merging (SBFOM) algorithm and Fuzzy FCA – Merge (FFCA-Merge) algorithm. The SBFOM algorithm is tested in an environment consisting of a set of fuzzy ontologies developed by different users for a particular domain. We merge these fuzzy ontologies and study how the number of new concepts learned and the number of similar concepts changes. For FFCA-Merge algorithm, which is an extension of FCA – Merge to fuzzy ontology merging, experiments have been conducted on synthetic data.

1. 5 Organization of the Thesis

Rest of the thesis is organized as follows. Chapter 2 discusses the related work, some of the tools and techniques for crisp ontology merging and some work related to fuzzy ontology learning. Chapter 3 describes the approaches that are followed to develop the algorithms that are used for fuzzy ontology merging. The pseudo codes of these algorithms are given and the implementations are also discussed. Chapter 4 is devoted for the experiments that are conducted to demonstrate the performance of both the algorithms. Chapter 5 concludes with the summary of the thesis.

Chapter 2

Related Work

Fuzzy ontology is a very new research area related to ontologies. Various methods have been developed for fuzzy ontology learning. In this chapter we concentrate on various tools and techniques for crisp ontology merging that have been developed, and also discuss some methods for fuzzy ontology learning.

2. 1 Ontology Merging Tools and Techniques

Ontology merging is required when we have two source ontologies and we want to build a new ontology that contains information from both the ontologies. It is also required when we want to merge new knowledge which is in the form of ontology into already existing ontology. To answer queries across different web sites having different ontologies ontology merging is required.

There are many methods, tools and techniques for crisp ontology merging. Researchers investigated the problem of designing the algorithms and tools for automatic ontology merging. But complete automation of this process is still not possible. In principle we can perform two types of ontology integration: *concept-level integration* and *syntactical-level integration.* Concept-level integration requires inference over the domain ontology to make the decision about the integration of a particular pair of concepts. Syntactical integration defines the rules in terms of the class and attributes names used in the ontologies to be integrated. Such integration rules are conceptually blind and are easy to develop and implement.

A first approach for supporting the merging of ontologies is described by Hovy (1998). Several heuristics are described for identifying corresponding concepts in different ontologies, e.g. comparing the names of two concepts, comparing the natural language definitions of two concepts by linguistic techniques, and checking the closeness of two concepts in the concept hierarchy [Stumme and Maedche, 2001].

2. 1. 1 PROMPT

PROMPT is an ontology management tool suite. The **PROMPT** suite includes an ontology merging tool (**iPROMPT**, formerly known as **PROMPT**). **iPROMPT** is an interactive ontology merging tool, which helps users in ontology merging task by providing suggestions about which elements can be merged, by identifying inconsistencies and potential problems and suggesting possible strategies to resolve these problems and inconsistencies [Noy and Musen, 2003].

2. 1. 2 Anchor-PROMPT

Anchor-PROMPT takes a set of anchors (pairs of related terms) from the source ontologies and traverses the paths between the anchors in the source ontologies. It compares the terms along these paths to identify similar terms and generates a set of new pairs of semantically similar terms [Noy and Musen, 2003].

2. 1. 3 OntoMorph

Chalupsky [Chalupsky, 2000] has developed **OntoMorph** which provides a powerful rule language for specifying mappings, and facilitates ontology merging and the rapid generation of knowledge-base translators. It combines two powerful mechanisms for knowledge-base transformations such as syntactic rewriting and semantic rewriting. Syntactic rewriting is done through pattern-directed rewrite rules for sentence-level transformation based on pattern matching. Semantic rewriting is done through semantic models and logical inference.

2. 1. 4 FCA - Merge

Stumme and Maedche [Stumme and Maedche, 2001] have developed **FCA-Merge** for merging ontologies based on *Formal Concept Analysis*. The **FCA-Merge** approach is a bottom-up approach, which means that it is based on application-specific instances of the two ontologies that need to be merged. A set of documents that are relevant to both ontologies are provided as input. Using linguistic analysis, instances are extracted from the documents for both ontologies. Now a pruned concept lattice is generated using the similarity in instances for both ontologies. These first two steps (lexical analysis and generating the concept lattice) are carried out fully automatically. In the third and last step of the method, the merged target ontology is created interactively (i.e. semi-automatically).

2. 1. 5 Chimaera

Chimaera is an ontology merging and diagnosis tool described in [McGuinness, et al, 2000]. Merging ontologies is to combine two or more ontologies that may use different vocabularies and may have overlapping content. The major two tasks that **Chimaera** performs are: (1) to coalesce two semantically identical terms from different ontologies so that they are referred to by the same name in the resulting ontology and (2) to identify terms that should be related by subsumption, disjointness, or instance relationships and provide support for introducing those relationships.

2. 1. 6 ONION

ONION [Mitra, et al, 2000] uses a graph-oriented model for the representation of the ontologies. In **ONION** there are two types of ontologies, source ontologies and articulation ontologies, which contain the concepts and relationships expressed as articulation rules. The mapping between ontologies is executed by ontology algebra. Such algebra consists of three operations, namely, intersection, union and difference. The intersection produces an ontology graph, which is the intersection of the two source ontologies, the union operator generates a unified ontology graph comprising the two original ontology graphs connected by the articulation, and difference operator is to distinguish the difference between two ontologies.

2. 2 Fuzzy Ontology Learning Methods

Many people have contributed for the work of generating a fuzzy ontology from the imprecise and uncertain data. Some of them are discussed below.

Tho *et al.* [Tho, et al, 2006] have developed a method for the automatic generation of the fuzzy ontology. In the literature there are methods for generation of ontologies from formal concept analysis. Ontology constructed by formal concept analysis is quite complicated in terms of the number of concepts generated and can not deal with the vague and uncertain information. So these authors have introduced fuzzy logic into the formal concept analysis to handle vague and uncertain data. They have obtained a fuzzy formal context from the given domain specific documents. They are using α*-cut* to eliminate relations with low membership values. From that fuzzy formal context by using fuzzy formal concept analysis (FFCA) they have obtained a fuzzy concept lattice. Objects that have small differences in terms of attribute values are classified into distinct formal concepts. Since such objects should belong to the same concept, they have used fuzzy conceptual clustering to cluster similar concepts. After that they have constructed hierarchical relations among conceptual clusters to obtain a concept hierarchy which is then used for fuzzy ontology generation.

Zhou *et al.* [Zhou, et al, 2007] have also suggested a method for fuzzy ontology generation. They have also used fuzzy formal context to deal with imprecise and uncertain data. In the fuzzy formal context they used different windows (α *-cut)* for different attributes. That means each attribute considers the relations having membership value in the range specified by the respective window and every attribute has a different window.

2. 3 WordNet

WordNet [Miller, 1995] is an online lexical database. It is designed for use under program control and provides effective combination of traditional lexicographic information and modern computing. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexical concept. Semantic relations link the synonyms sets. In this thesis we are using the WordNet as the lexical database or thesaurus that gives the relations between terminological terms.

Chapter 3

Fuzzy Ontology Merging Algorithms

In this chapter we have developed two algorithms for fuzzy ontology merging. The first algorithm called Similarity Based Fuzzy Ontology Merging (SBFOM) algorithm, is based on the similarity measure to find the similarities between the concepts of fuzzy ontologies. Detailed discussion about internal and external representation of fuzzy ontologies, the SBFOM algorithm, its pseudo code, its implementation, and a brief description about important functions is presented in the first section. In the second section we discuss how FCA-Merge is extended to develop fuzzy ontology merging (FFCA – Merge) algorithm. Also, a detailed discussion about the FFCA – Merge algorithm, its pseudo code, its implementation, and a brief description about important functions is presented.

3. 1 Similarity Based Fuzzy Ontology Merging Algorithm

There are two important goals behind the development of this algorithm. The first one is to develop an algorithm that is suitable for both crisp ontology merging and fuzzy ontology merging. The second one is to develop an automatic merging algorithm for ontologies. This can be achieved by providing the information which is required from the user in the form of a thesaurus which is built from the WordNet.

This algorithm takes two source fuzzy ontologies as input and returns a merged fuzzy ontology as output. We consider the first fuzzy ontology as the existing merged fuzzy ontology and the second fuzzy ontology as the source fuzzy ontology which is merged into the first fuzzy ontology. That means, the first fuzzy ontology (existing merged fuzzy ontology) is extended with new concepts and relations from the second fuzzy ontology (source fuzzy ontology). We can say that for a particular domain this algorithm can also be used for incrementally inserting new knowledge which is in the form of fuzzy ontology into the existing merged fuzzy ontology (first fuzzy ontology) to make it fully aware of the domain knowledge. In this way we are actually building a global fuzzy ontology for that domain.

3. 1. 1 Representation of Fuzzy Ontologies

In this subsection we describe what a fuzzy ontology is and how we represent a fuzzy ontology. A fuzzy ontology can be viewed as a knowledge base consisting of various objects and weighted relationships among those objects. These objects are concepts or classes. We are using XML to represent fuzzy ontologies. We are using an internal representation also. But this internal representation is invisible to the users. In this internal representation fuzzy ontology is represented as a list of nodes. Each node is having the name, which is the corresponding concept's name, and a list of arcs, which are relations along with weights from this concept to other concepts of the fuzzy ontology. This representation can be visualized as a graph structure and all required operations can be performed on it. This internal representation is very useful because we can add new nodes easily if we want to add new concepts to the existing merged fuzzy ontology.

3. 1. 2 Merging Similar Concepts

When we are merging two fuzzy ontologies the first thing we have to do is to find matching concepts. To find similarity between two concepts we need a similarity measure. The similarity measure we are using in this algorithm consists of three main components. They are linguistic similarity, contextual similarity, and lexical similarity. The main motivation behind this composition of the similarity measure is, in a particular domain if two concepts are similar then there is a greater possibility for them to have similar contexts also (contextual similarity). And it is also possible that the designers who build ontologies tend to use same name to represent the similar concepts (lexical similarity). Even if they use different terms to represent the same concept in a domain, meaning of those terms should be same (linguistic similarity). So by combining these three components most similar concepts from both fuzzy ontologies will be merged. The similarity measure used in this algorithm is defined as

$$\text{Sim} = w_1 * \text{Li\_Sim} + w_2 * \text{C\_Sim} + w_3 * \text{Le\_Sim}$$

where w_1, w_2 and w_3 are weights of linguistic similarity, contextual similarity and lexical similarity in the similarity measure respectively.
We explain each of these components in the following sections.

Lexical Similarity

As we mentioned above in a particular domain ontology designers tend to use the same name to represent two similar concepts. So while finding similarity between two concepts it is reasonable to give some weight to their lexical similarity. Lexical similarity can be defined in many ways. Substring similarity, Levenshtein similarity measure to find edit distance are some examples of the lexical similarity measure.

In this algorithm we use substring similarity to find the lexical similarity between two concepts. This substring similarity of two concepts is defined as the length of the longest substring of the given concepts divided by the length of longest string among the two input concept names. For example "Animal" and "Animals" are the two input strings. Their longest substring is "Animal" of length 6. And "Animals" with length 7 is the longest among the two input strings. So their substring similarity is defined as 6/7 that is 0.8571.

Linguistic Similarity

In a particular domain it is possible that same concept is represented by different names. That means their lexical similarity is almost zero. So by using only syntactic (lexical) similarity we can not find all similar concepts from both fuzzy ontologies. For example in Figure 3.1 concept named "Person" of fuzzy ontology 1 is having "Living" as parent and many children and in fuzzy ontology 2 concept named "Human" is having "Organism" as parent and many children. By using only lexical similarity measure we cannot declare that these two concepts are similar. To handle this type of cases we have made semantic similarity as a component of our similarity measure and we are using a thesaurus (WordNet) to find semantic similarity. In the above example by using thesaurus to find the semantic similarity of given concepts which is now a part of our similarity measure, we can conclude that "Person" and "Human" are equivalent and similarly we can declare that "Living" and "Organism" are also represent the same concept.

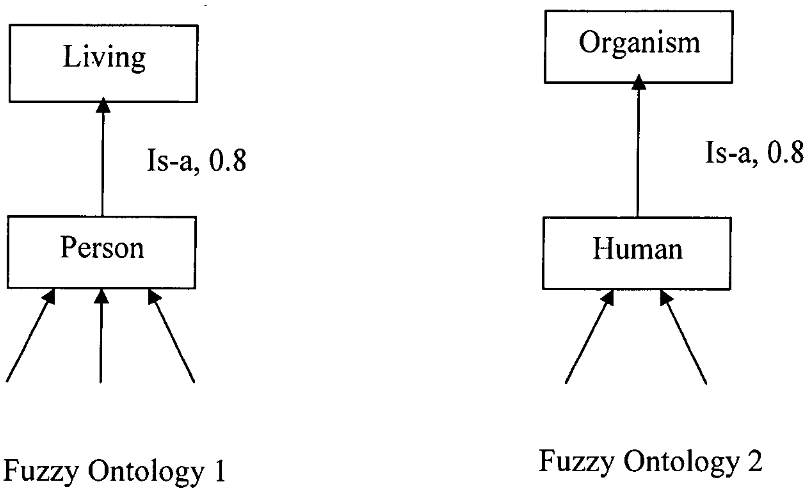


Figure 3.1. Two fuzzy ontologies to explain Importance of Linguistic Similarity

Linguistic similarity is the similarity of the semantic content of the names of two concepts. To obtain linguistic similarity we are using the network of terminological relationships represented by a thesaurus. A thesaurus of

TH-17486
006'3
L1499
Fw

terminological relationships can be built by exploiting WordNet which is a source of lexical information. In [Castano, et al, 2003] linguistic similarity of two concepts is defined as the highest strength path of terminological relationships between two concepts in the thesaurus if at least one path exists and zero otherwise.

Contextual Similarity

In [Castano, et al, 2003] context of a concept is defined as union of set of adjacent concepts and set of properties of that concept. For example in the Figure 3.2, part of the fuzzy ontology shown inside the rectangle represents the context of the concept "Publication". This context consists of the concept "Book" along with the relation "same-as", the concept "Periodical" along with the relation "is-a", and the properties "Publisher" and "Title" along with their relations "has".

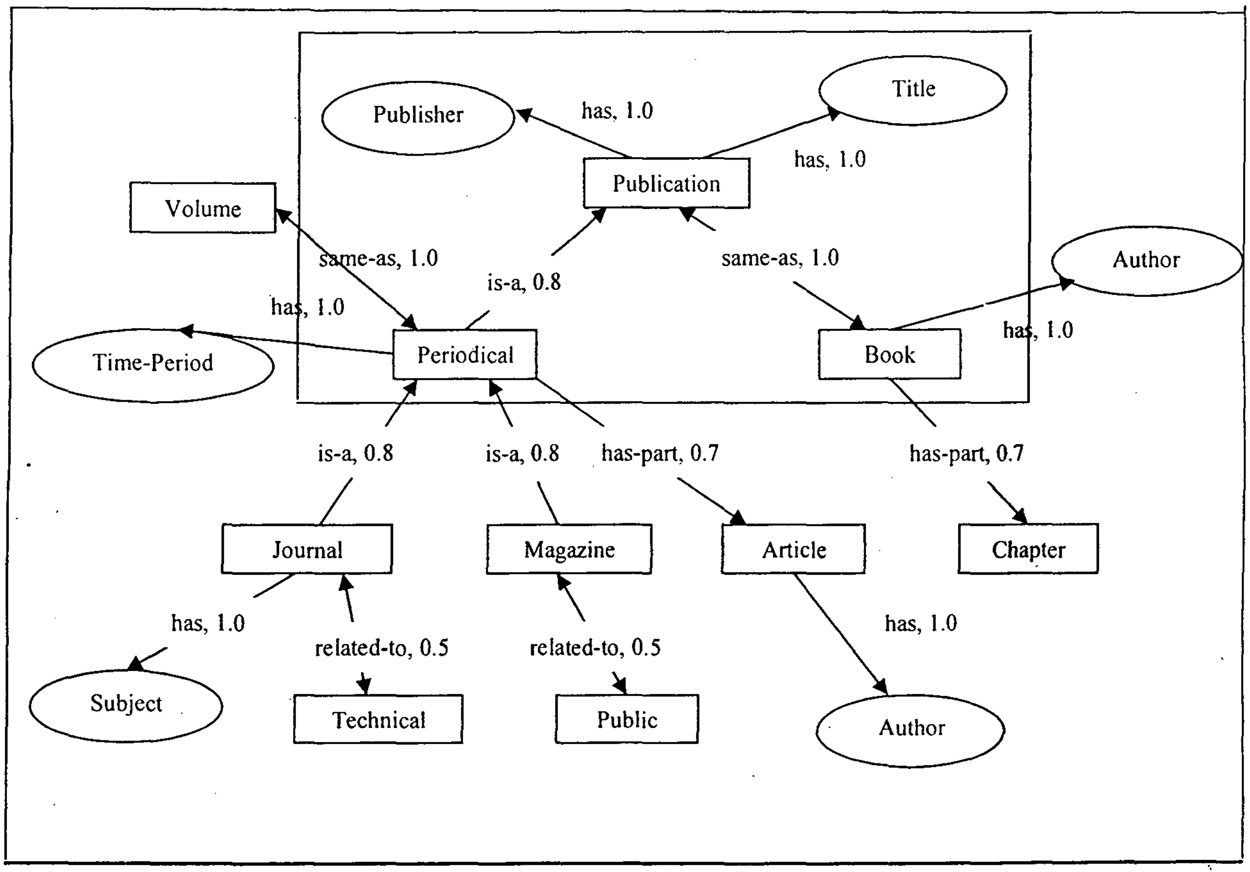


Figure 3.2. Context of Publication concept of the ontology taken from [Castano, et al, 2003].

In general the context of a concept **C** can be represented by a context vector **Context (C)** = $\{cv_1, \ldots cv_n\}$, where $\forall i \in (1, \ldots n)$, $cv_i = (f_i, r_i, w_i)$, where $\mathbf{f}_i$ denotes either an adjacent concept or a property of **C**, r_i denotes the semantic relation between **C** and f_i, and w_i denotes the weight associated with the semantic relation between **C** and f_i. So

Context (Publication) = {(Book, same-as, 1.0), (Periodical, is-a, 0.8),
(Publisher, has, 1.0), (Title, has, 1.0)}

Contextual similarity is the similarity of contexts of two given concepts. In a particular domain if two concepts are similar their contexts should also be similar. To find contextual similarity between two concepts first we have to obtain contexts in the form of vectors as shown above, of given two concepts. After obtaining those vectors, for all pairs of terms from two context vectors we have to obtain product of linguistic similarity of properties and adjacent concepts and closeness of semantic relations, and summation of those products normalized by the product of lengths of two vectors will give us the contextual similarity.

To find the source fuzzy ontology concepts those can be merged with the existing merged fuzzy ontology concepts, for a source fuzzy ontology concept, we find the existing merged fuzzy ontology concept that is having highest similarity and that similarity must be greater than the predefined threshold for similarity. If we could find such a concept from the existing merged fuzzy ontology, we will merge the source fuzzy ontology concept with that concept, and this merged concept will be added to a list merged concepts for further processing. This same process will be repeated for the remaining source fuzzy ontology concepts. At the end of this process we will be having a list of merged concepts. For every merged concept from the list we have to execute the processes given in the following subsections.

3. 1. 3 Updating Membership Values

Now we explain how the weights of the relationships change as a result of merging similar concepts through an example. When we merge two fuzzy ontologies as shown the Figure 3.3, the “Book” concepts from both the fuzzy ontologies as well as the “Publication” concepts are merged.

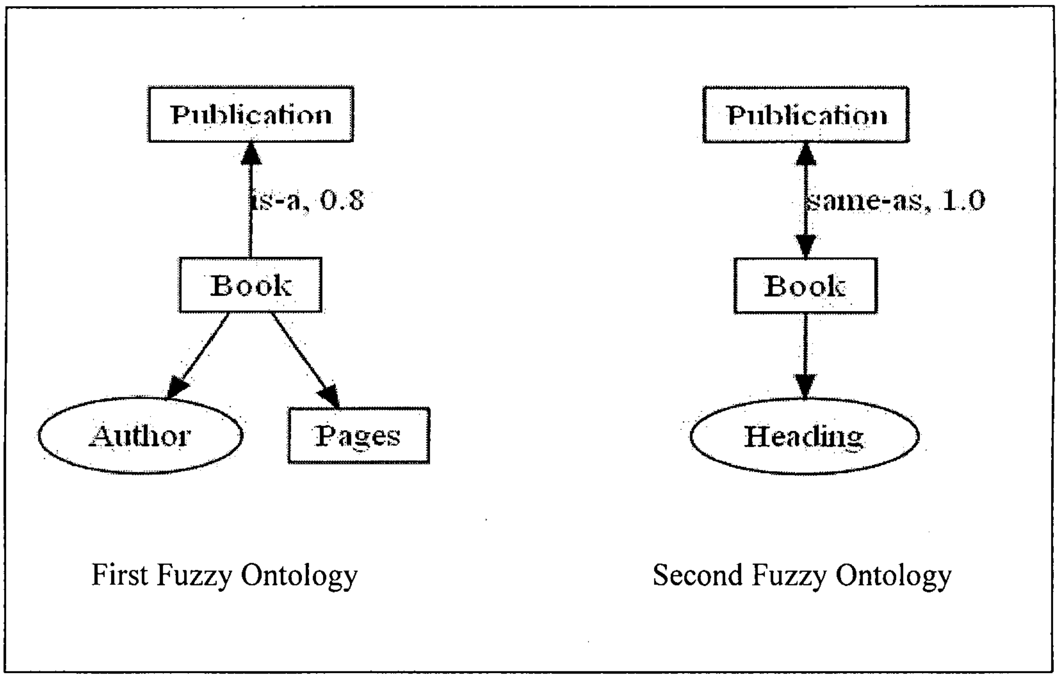


Figure 3.3. Two fuzzy ontologies to explain changing relation membership values

First fuzzy ontology says that “Book” is sub concept of (0.8) “Publication” where as second says that “Book” and “Publication” are equivalent (1.0). After considering these two relations and their weights (0.8 and 1.0) we will assign the average (0.9) of two weights as the membership value of the relationship between these two merged concepts in the existing merged fuzzy ontology. So for all merged concepts if their super concepts have also got merged then we have to update membership values as explained in this section. Super concepts are those concepts to which the present concept is directly related and sub concepts are those concepts which are directly related to the present concept.

3. 1. 4 Discovering New Concepts and Relations

After updating membership values the next step in this merge algorithm, is "**discovering new concepts and relations**". An objective of this step is to discover new concepts and add them to the existing merged fuzzy ontology. We use the term "new concepts" to represent those concepts which are known to the source fuzzy ontology and not yet understood by the existing merged fuzzy ontology.

For example, consider the fuzzy ontologies shown in the Figure 3.4. As result of merge process "Multicellular" concept of ***FO_2*** got merged with the "Multicellular" concept of ***FO_1***. When we merge these two concepts we also copy relations from the concept "Multicellular" to its superconcepts (that is "Living") to ***FO_1***. But the concept to which these relations refer to may not present in the merged fuzzy ontology. These relations have become dangling. To avoid problems of this type and not to loose any concepts from ***FO_2*** we copy all non-merged super concepts of a merged concept from ***FO_2*** to ***FO_1***. So we copy

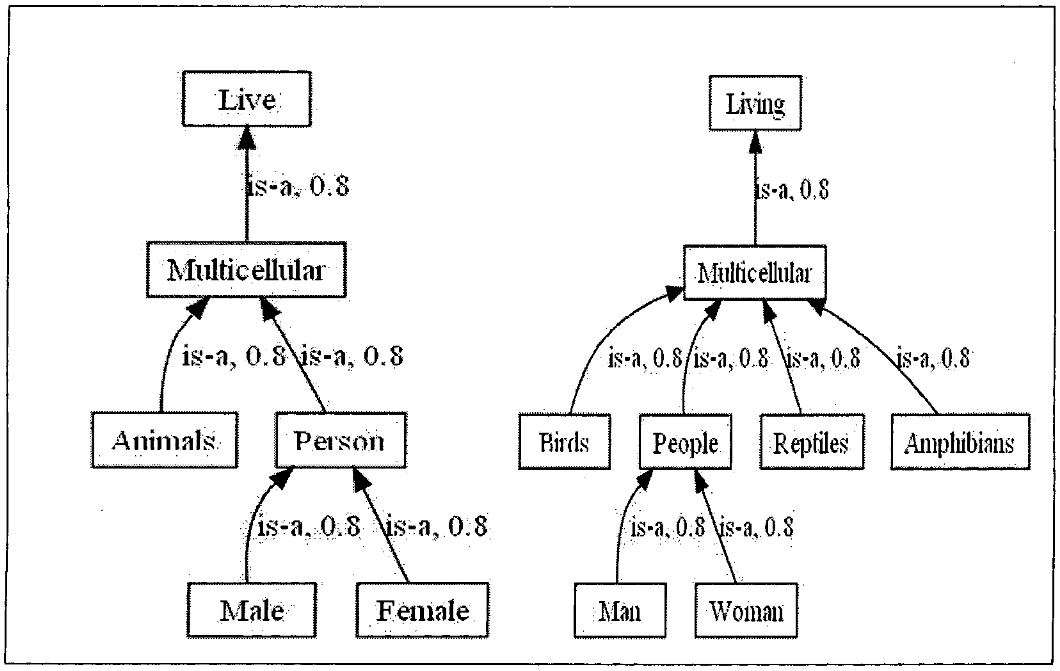


Figure 3.4. Fuzzy Ontologies ***FO_1*** and ***FO_2***

the concept "Living" to $\boldsymbol{FO_1}$ and this process is repeated for all merged concepts and concepts copied from the source fuzzy ontology. Similarly as a result of performing the merge, relation from the concept "Birds", (which is the sub concept of "Multicellular") to the concept "Multicellular" has become dangling. Because of the above mentioned reason we copy all non-merged sub concepts of a merged concept from $\boldsymbol{FO_2}$ to $\boldsymbol{FO_1}$. While discovering and adding new concepts from $\boldsymbol{FO_2}$ we also copied relations of them to the existing merged fuzzy ontology ($\boldsymbol{FO_1}$). Relations known to $\boldsymbol{FO_2}$, and copied to $\boldsymbol{FO_1}$ are called discovered relations.

After discovering new concepts and relations that are known to other fuzzy ontology, we can identify new relations between discovered concepts and concepts already present in the existing merged fuzzy ontology with the help of WordNet. We can identify new relations like "Hypernym" and "Hyponym". "Hypernym" identifies super concepts of the given concept. For example "Animal", is hypernym of "Bird". "Hyponym" identifies sub concepts of the present concept and is reverse of "Hypernym". For example "Hen" is hyponym of "Bird". These two provides us support in identifying "is-a" relations. "Holonym" is the other relation provided by the WordNet. "Holonym" provides us the whole of which the given concept is part of. For example "Forest" is holonym of "Tree". Reverse of this relation is "Meronym". This provides concepts which are parts of the given concept. For example "Tree Trunk" is meronym of "Tree". These two provides us support in identifying "has-part" relations.

Other relationship provided by the WordNet that we plan to use is "Synonym". This provides equivalent concepts of the given concept. For example "SingleCellular" is synonym of "UniCellular". This helps us in establishing equivalence relations between discovered concepts and concepts already known to $\boldsymbol{FO_1}$. For example, while merging fuzzy ontologies as shown in the Figure 3.4

we can learn and add an equivalence relation between concepts "Female" and "Woman" and many relations can be learnt.

After considering all super concepts and sub concepts of a merged concept C'_C, where C' is the existing merged fuzzy ontology concept with which the source fuzzy ontology concept C got merged, we copy the properties of C as the properties of C'_C if they are not already present. In the same way remaining concepts from the merge list will be explored. At the end of this process we will be having the merged fuzzy ontology in the form of existing merged fuzzy ontology with extended knowledge.

3. 1. 5 Algorithm – I (SBFOM Algorithm)

We describe the Similarity Based Fuzzy Ontology Merging (SBFOM) Algorithm that is used for the entire process.

i. This algorithm takes two fuzzy ontologies as input.
ii. One of the two fuzzy ontologies is taken as the existing merged fuzzy ontology and the second one as the source fuzzy ontology which will be merged into the existing merged fuzzy ontology.
iii. To find matching concepts we have to find similarity among the concepts of both fuzzy ontologies
iv. Similarity measure used in this algorithm consists of three components. They are linguistic similarity, contextual similarity, and lexical similarity.
    a. To find linguistic similarity use the thesaurus which is built from the WordNet.
    b. Find out similarity of contexts of the two given concepts.
    c. Use substring similarity measure as lexical similarity of given concepts.
    d. Weighted sum of these components will give us similarity value of given concepts.

v. A concept from the source fuzzy ontology will be merged with the existing merged fuzzy ontology concept that is having the highest similarity and that similarity must be greater than the defined threshold. This merged concept is added to a list of merged concepts for further processing. This procedure is repeated for every concept of the source fuzzy ontology to find all matches.

vi. After performing all possible merges, for every merged concept $C^{'}\_C$ (here $C^{'}$ is the existing merged fuzzy ontology concept and C is the source fuzzy ontology concept) from the merge_list we have perform the following procedure.

   a. If a super concept of C got merged with a super concept of $C^{'}$ adjust the membership value of the relation between $C^{'}\_C$ and its merged super concept as average of the membership value of the relation between $C^{'}$ and its super concept and the membership value of the relation between C and its super concept. If required repeat this step for other superconcepts of C.

   b. For all super concepts of C those got merged but not with the super concepts of $C^{'}$, copy the relation between C and its super concept as relation between $C^{'}\_C$ and that merged super concept along with the membership value.

   c. All remaining non-merged super concepts of C will be copied as super concepts of $C^{'}\_C$ along with relations and membership values. And add those concepts to the list of merged concepts for further processing. Identify and add new relations with the help of thesaurus.

vii. Copy non-merged sub concepts of C as sub concepts of $C^{'}\_C$ along with their relations and membership values. And add those concepts to the list of merged concepts for further processing. Identify and add new relations with the help of thesaurus.

viii. Copy properties of C as properties of $C^{'}\_C$.

At the end of this process the existing merged fuzzy ontology is augmented with new concepts and relations from the source fuzzy ontology.

Pseudo code of Algorithm - I

In the Figure 3.5 the pseudo code of the algorithm is presented. This gives details about how the algorithm is implemented.

```
Input: $FO_1$,$FO_2$ are two input fuzzy ontologies
Output: $FO_1$ augmented with new concepts and relations from $FO_2$
Process:
        Concept: C, M_List[], C', Merge_Prob, C'_C, Ctx1, Ctx2;
        Float: S_Max, L_Sim, C_Sim, S_Sim, S;
        Begin
            For each concept C belongs to $FO_2$ do begin
                Ctx1 = obtain_context(C);
                S_Max =Threshold;
                For each concept C' belongs to $FO_1$ do begin
                    L_Sim = Find linguistic similarity of C and C';
                    C_Sim = Find contextual similarity of C and C';
                    S_Sim = Find lexical similarity of C and C';
                    S = (1/3) * L_Sim + (1/3) * C_Sim + (1/3) * S_Sim;
                    If(S > S_Max) then begin
                        S_Max = S;
                        Merge_Prob = C';
                    End If
                End For
                If (Merge_Prob is not Null) then begin
                    C'_C = merge C' and C;
                    Add(M_List,C'_C);
                End If
            End For
            If (M_List is not empty) then begin
                For each merged concept C'_C do begin
                    For all super concepts of C
                        If (supers_merged(C',C)) then begin
                            adjust relation_weight( merged concept, super concept);
                        Else if(super_merged(C)) then
                            add new relation;
                        Else
                            add new concept to existing merged fuzzy ontology
                            and add the same to Merge_list;
                    End If; End For all;
                End For
                For each non merged sub concept of C do begin
                    add new concept to existing merged fuzzy ontology
                    and add the same to Merge_List;
                End For
            Else
                Exit;
            End If
    End Process;
```

Figure 3.5. Pseudo code of the Algorithm - I

3. 1. 6 Implementation

Some important functions involved in the implementation of similarity based fuzzy ontology merging algorithm are explained below. Also we mention about their inputs, outputs and working details.

xmltograph

This function takes XML document of a fuzzy ontology as an input. It is used to obtain the textual representations from the given XML representations of the input fuzzy ontologies. We are using a fixed format to represent fuzzy ontologies using XML. This function parses the given XML file and writes information such as number of nodes, names of nodes, number edges, and edges between nodes to a text file. After reading that text file we are generating the graphical representation of the given fuzzy ontology in the main module itself.

path_length

This function takes two concepts as input and returns semantic similarity of them. Semantic similarity is computed as the product of weights of edges on the longest path in the thesaurus between given concepts. This function mainly computes the longest path between given concepts in the thesaurus and obtains the product of weights of edges on that path as linguistic similarity. This function is called by **Sim** function. Thesaurus is built with the help of WordNet.

sim

This function takes two concepts as input and returns their similarity. It calls **sub_sim** function to find the lexical similarity of given concepts. Then it calls the function **path_length** to find their linguistic similarity. It also obtains contexts of given concepts and finds their similarity. Then it returns the weighted sum of these three values as similarity of given two concepts.

sub_sim

This function also takes two concept names as input. It finds the length of longest substring of them. Then it divides that length with the length of the longest string (concept name), and returns the resulting value as lexical similarity of given concept pair.

merge

This is one of the main modules of our implementation. This function merges matching concepts from given fuzzy ontologies. To find a matching concept from the existing merged fuzzy ontology for a given source fuzzy ontology concept, it calls **sim** function. A source fuzzy ontology concept is merged with the most similar existing merged fuzzy ontology concept. After finding all matches it will merge them and keeps track of merged concepts in the form of a linked list.

Graphviz Tool

Graphviz [Graphviz, 2004] is an open source graph visualization software. It has got several graph layout programs. It automatically draws diagrams based on the structural information of the graph given in the textual representation. All diagrams shown in this dissertation are drawn with the help of this tool. Graph descriptions are stored in dot files and when we run those dot files we will obtain their diagrams.

graphtodot

This function is mainly used to convert the resulting merged fuzzy ontology into its graphical representation. It converts the internal representation of a fuzzy ontology into the dot representation (dot file). If we run this dot file by using Graphviz tool we can obtain graphical visualization of the existing merged fuzzy ontology.

3. 2 Fuzzy FCA - Merge Algorithm

In this algorithm, we extended a crisp ontology merging method FCA-Merge to fuzzy ontology merging and is called as FFCA – Merge Algorithm. This method uses fuzzy formal concept analysis. According to [Ganter and Wille, 1999], formal concept analysis studies how objects can be hierarchically grouped together according to their common attributes. One of the aspects of FCA thus is attribute logic, the study of possible attribute combinations. This algorithm requires some theory related to fuzzy formal concept analysis.

3. 2. 1 Definitions

In [Tho, et al, 2006] **Fuzzy formal context** is defined as a triple K = (G, M, I = (G $*$ M)), where G is a set of objects, M is a set of attributes and I is a fuzzy set on domain G $*$ M. Each relation (g, m) $\in$ I has a membership value μ(g, m) in [0, 1].

Each **Object** O in a fuzzy formal context K can be represented by a fuzzy set $\phi(O)$ as $\phi(O) = \{A_1(\mu_1), A_2(\mu_2)$ - - -$A_m(\mu_m)\}$, where $\{ A_1, A_2$ - - - $A_m \}$ is the set of attributes in K and μ_i is the membership value of O with attributes A_i in K. $\phi(O)$ is called the fuzzy representation of O [Tho, et al, 2006].

Fuzzy formal concept has been defined as extension of formal concept. According [Tho, et al, 2006], given a fuzzy formal context K = (G, M, I) and a confidence threshold T we define $\boldsymbol{A}^* = \{m \in M \mid \forall g \in \boldsymbol{A} : \mu(g, m) \geq T\}$ for $\boldsymbol{A} \subseteq G$ and $\boldsymbol{B}^* = \{g \in G \mid \forall m \in \boldsymbol{B} : \mu(g, m) \geq T\}$ for $\boldsymbol{B} \subseteq M$. A fuzzy formal concept (or fuzzy concept) of a fuzzy formal context (G, M, I) with a confidence threshold T is a pair $(A_f = \varphi(A), \boldsymbol{B})$, where $A \subseteq G$, $\boldsymbol{B} \subseteq M$, $A^* = \boldsymbol{B}$, and $\boldsymbol{B}^* = A$.
Each object g $\in \varphi$(A) has a membership μ_g defined as

$$\mu_g = \min_{m \in B} \mu(g, m),$$

where μ (g, m) = membership value between object g and attribute m defined in I.

A is called extent and B is called intent of the fuzzy concept. The subconcept – superconcept relationship is formulated by

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2, (\Leftrightarrow B_1 \supseteq B_2)$$

The set of all concepts of a fuzzy formal context K together with the partial order $\leq$ is always a complete lattice, called the fuzzy concept lattice of K.

Fuzzy concept lattice is represented by a line diagram. In a line diagram every node represents a fuzzy formal concept. A concept C_1 is a subconcept of a concept C_2 if and only if there is a path of descending edges from the node representing C_2 to the node representing C_1. The name of an object g is always attached to the node representing the smallest concept with g in its extent; the name of an attribute m is always attached to the node representing the largest concept with m in its intent. The extent of a fuzzy formal concept is union of all objects whose labels are attached to subconcepts. The intent of a fuzzy concept is union of all attributes of all superconcepts.

3. 2. 2 Main Approach

In this approach we merge fuzzy ontologies in bottom_up fashion. The first step of this approach extracts instances (fuzzy contexts) of both source fuzzy ontologies from a given set of domain specific documents by applying natural language processing techniques. Then in the second step, we merge these two fuzzy contexts and obtain merged fuzzy context which is used to derive a lattice of concepts as a structural result of FFCA - Merge. In the third step, the produced fuzzy concept lattice is explored and transformed to the merged fuzzy ontology. Now we describe all the above three steps in detail.

Instance Extraction and Generation of FFC

Main goal of this step is to extract instances of two input fuzzy ontologies from the given domain specific documents. When a document contains an instance of a concept of a fuzzy ontology this fact is represented by a relation (g, m), where g is a document and m is a concept. This relation is also associated with a membership value. This membership value is calculated as the ratio of the number of instances of m in g and the number of instances of all attributes or concepts of that fuzzy ontology in g.

Given domain specific documents forms the object set and concepts of the fuzzy ontology becomes attribute set of the fuzzy formal context. These two sets and relations obtained as explained above arranged in the form of a table gives the fuzzy formal context of the given fuzzy ontology. The FCA – Merge method also obtains the contexts of given ontologies but it does not calculate membership values associated with relations.

For example consider two fuzzy ontologies shown in Figure 3.6. These two fuzzy ontologies are structural representations of various things related to the real world and relationships among those things. Concepts of a fuzzy ontology forms attribute set of the corresponding fuzzy formal context. So attribute sets of fuzzy formal contexts of two input fuzzy ontologies are

$\boldsymbol{M_1}$ = {Thing, NonLivingThing, Human, Animal, WildAnimal}

and

$\boldsymbol{M_2}$ = {Thing, LivingThing, People, Lion}.

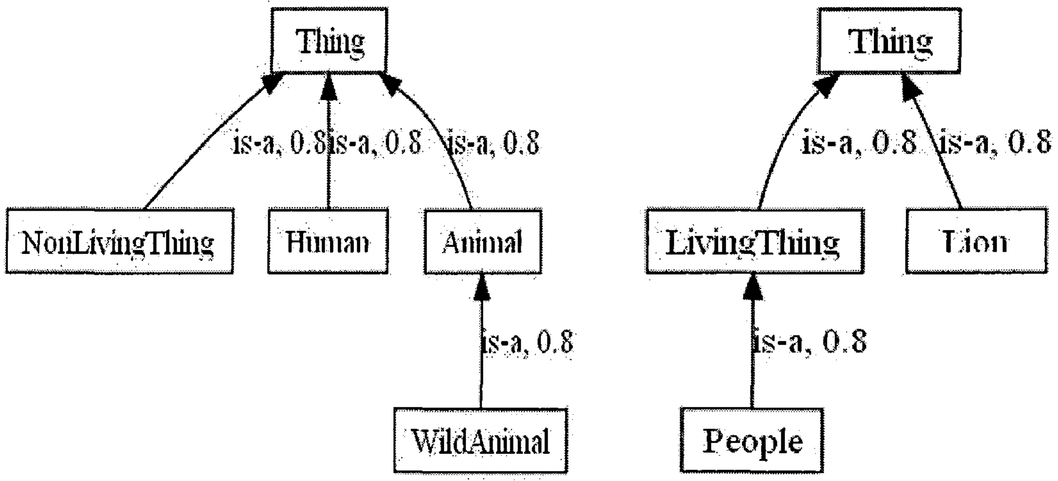


Figure 3.6. Two input fuzzy ontologies 1 and 2

	Thing	Human	NonLiving Thing	Animal	Wild Animal
D1	1	0.9	0.8	0.7	0.8
D2	1	0.9	0.8	0.8	0.7
D3	1	0.7	0.9	0.9	
D4	1	0.9	0.7	0.8	0.7
D5	1			0.7	0.8
D6	1	0.8		0.9	0.7
D7	1	0.7			
D8	1	0.8	0.7	0.9	0.7
D9	1	0.9	0.8	0.7	
D10	1	0.8	0.9	0.7	
D11	1	0.9	0.7	0.8	0.7
D12	1	0.8			
D13	1	0.7		0.8	0.9
D14	1	0.7	0.7	0.9	

	Thing	Living Thing	People	Lion
D1	1	0.9	0.9	0.8
D2	1	0.9	0.9	0.7
D3	1	0.7	0.7	
D4	1	0.9	0.9	0.7
D5	1			0.8
D6	1	0.8	0.8	0.7
D7	1	0.7	0.7	
D8	1	0.8	0.8	0.7
D9	1	0.9	0.9	
D10	1	0.8	0.8	
D11	1	0.9	0.9	0.7
D12	1	0.8	0.8	
D13	1	0.7	0.7	0.9
D14	1	0.7	0.7	

Table 3.1: Fuzzy Formal Context of Fuzzy Ontology 1

Table 3.2: Fuzzy Formal Context of Fuzzy Ontology 2

Suppose there are 14 documents that are containing information about this domain. These documents forms the object set of both fuzzy formal contexts. After extracting instances of both fuzzy ontologies, their fuzzy formal contexts are shown in the Tables 3.1 and 3.2. We generated this data artificially to explain our algorithm using α -cut (with α = 0.5) to eliminate relations with low membership values. The extraction of the instances from documents is necessary because there are usually no instances which are already classified by both ontologies. However, if this situation is given, one can skip the first step and use the classification of the instances directly as input for the two formal contexts [Stumme and Maedche, 2001].

Fuzzy Concept Lattice Generation

We merge the fuzzy formal contexts obtained in the previous step. From the merged fuzzy formal context we derive the fuzzy concept lattice. Here note that the same concept may present in both fuzzy ontologies but treated differently. To make same concepts from different fuzzy ontologies distinct we perform concept disambiguation on the obtained fuzzy formal contexts. The concept disambiguation is to attach to each concept the fuzzy ontology number to which that concept belongs to.

Let $M_i' = \{(m, i) \mid m \in M\}$, for $i \in 1, 2$. Then the merged fuzzy formal context is obtained by $K = (G, M, I)$ with $G = D$, $M = M_1' \cup M_2'$, and $(g, (m, i)) \in I \Leftrightarrow (g, m) \in I_i$. After performing concept disambiguation we merge both fuzzy contexts and obtain the merged fuzzy context [Stumme and Maedche, 2001].

Next, the computation of pruned concept lattice is done with the modified TITANIC algorithm. When compared to other algorithms for computing concept lattices, TITANIC has the advantage that it computes the fuzzy formal concepts via their key sets or minimal generators [Stumme and Maedche, 2001]. A key set is minimal set of attributes which generates a given fuzzy formal concept.

In [Stumme, et al, 2000] authors have used the composed function $.'' : \mathcal{B}(M) \rightarrow \mathcal{B}(M)$ which is a closure operator on M (that is extensive, monotonous, and idempotent). The related closure system (that is the set of all $B \subseteq M$ with $B'' = B$) is exactly the set of the intents of all concepts of the context. The structure of the fuzzy concept lattice is already determined by this closure system. The computation of the closure system makes extensive use of a support function which is compatible with the closure operator. We have changed this support function to make it applicable for the fuzzy concept lattice computation. The support of $X \subseteq M$ is defined by

Supp(X) = (Sum of X's membership values of objects in the extent of X)/|G|
If cardinality of X is 1, else
Supp(X) = (Sum of x_i's membership values of objects of in the extent of X)/|G|
where $x_i \in X$ and x_i is the attribute having minimum support (weight) in X.

By using this modified support function in the TITANIC algorithm we can obtain fuzzy formal concept lattice. We compute those fuzzy formal concepts which are above at least one fuzzy formal concept generated by a fuzzy ontology concept of the source fuzzy ontologies as specified in FCA – Merge. The Figure 3.7 shows the fuzzy concept lattice obtained for our example by using modified TITANIC algorithm. Weights assigned to edges represent the similarities of respective fuzzy formal concept pairs [Tho, et al, 2006].

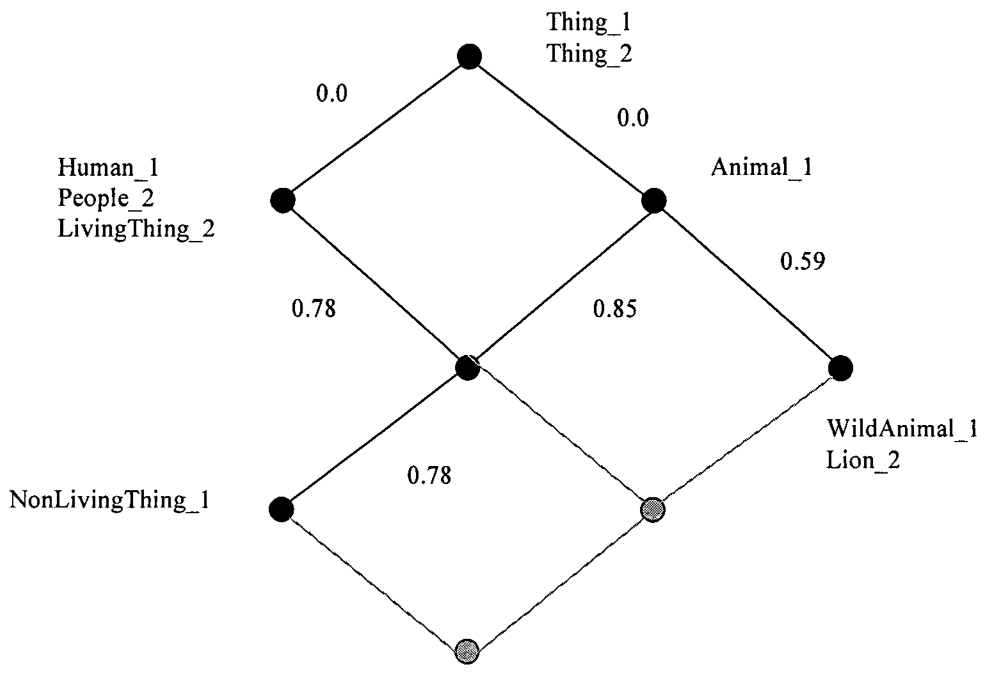


Figure 3.7. Fuzzy concept lattice

Obtaining Merged Fuzzy Ontology

We derive the target fuzzy ontology from the pruned fuzzy concept lattice which is the result of the last step. While generating the merged fuzzy ontology we consider only fuzzy formal concepts having key sets of cardinalities 1 and 2. This allows us to retain all concepts from the two source fuzzy ontoligies and to discover binary relation between concepts from the different source fuzzy ontologies, but no relation of higher arity. Each of the fuzzy formal concepts of the pruned concept lattice is a candidate for a concept, a relation, or a new subsumption in the target fuzzy ontology. For each fuzzy formal concept of the pruned concept lattice we will analyze the related key sets which are sets of source fuzzy ontology concepts to generate merged fuzzy ontology concepts.

In the FCA- Merge method [Stumme and Maedche, 2001] the details are given to generate merged ontology concepts from formal concepts, but not specified how to copy relations from source ontologies to the merged ontology. To generate concepts of the merged fuzzy ontology we followed the similar procedure given in FCA – Merge method. Since no procedure is given for copying relations from source fuzzy ontologies to the merged fuzzy ontology we developed our own procedure. Details are given in the following paragraphs.

We make fuzzy formal concept having empty set as a key set as the root of the merged fuzzy ontology. In the fuzzy concept lattice shown in the Figure 3.7 fuzzy concept labeled with {Thing_1, Thing_2} will be root of our merged fuzzy ontology. Next we copy the fuzzy formal concepts having a single key set of cardinality one as the merged fuzzy ontology concepts. Fuzzy concepts labeled with key sets {NonLivingThing_1} and {Animal_1} in the Figure 3.7 are also copied to the merged fuzzy ontology.

Fuzzy formal concept having more than one key set of cardinality one is generated by two or more concepts of the source fuzzy ontologies. This indicates that these concepts should be merged into one concept in the target fuzzy ontology. We generate merged concepts as suggested. In our example fuzzy formal concept lattice shown in the Figure 3.7 fuzzy concepts labeled with key sets {Human_1, People_2, LivingThing_2} and {WildAnimal_1, Lion_2} are candidates for merging. In FCA – merge this case is handled by either knowledge engineer can directly merge these concepts or can take advice of the user. We are not taking any input or advice from the user. We directly follow the suggestions of the lattice and merge those concepts from input fuzzy ontologies.

Next we explore the fuzzy formal concepts generated by at least two concepts from the source fuzzy ontologies and are candidates for new relations in the target fuzzy ontology. So we add new relations to the target fuzzy ontology between concepts in the key set (of cardinality 2). In this case also we follow the suggestions of the lattice and we do not take any input from the user. The fuzzy concept in the middle of the fuzzy concept lattice shown in the Figure 3.7 has key sets {Human_1, Animal_1}, {People_2, Animal_1}, and {LivingThing_2, Animal_1}. If a relation does not exist between these concept pairs we will establish relations between them with weight 0.5 (since these concepts are related and we do not know the strength of the relation). As mentioned before we will not consider fuzzy formal concepts having key sets of cardinality more than 2.

Now all concepts from the source fuzzy ontologies are included in the target fuzzy ontology. And new relations suggested by key sets of cardinality 2 are also generated in the merged fuzzy ontology. Now we have to copy all relations from the two source fuzzy ontologies to the merged fuzzy ontology. For that purpose we explore the merged fuzzy ontology concepts one by one, and we start with the root of the merged fuzzy ontology.

Suppose C1-C2 is a merged fuzzy ontology concept, where C1 is a concept of the first fuzzy ontology and C2 is a concept of the second fuzzy ontology. We use the term participants to represent the concepts (C1, C2) who have been merged to generate a merged concept (C1-C2). We find merged fuzzy ontology concepts consisting sub concepts C1. Suppose S1-S2 is a merged fuzzy ontology concept and S1 is a sub concept of C1. So we copy the relation between C1 and S1 along with the membership value as a relation between C1-C2 and S1-S2. If a relation is already established between these two merged concepts we will adjust the weight of the relation as the average of weight of the existing relation and weight of the incoming relation (that is between C1 and S1). This procedure is repeated for all subconcepts of all participants (C1, C2) of a merged concept (C1-C2).

If the merged fuzzy ontology concept C is not a merged concept, then we find the merged fuzzy ontology concept S' having sub concept of C as a participant. Then we copy the relation between C and its sub concept as the relation between C and S' along with the membership value. This procedure is repeated for all subconcepts of C. In the same way we explore other merged fuzzy ontology concepts. At the end we have resultant merged fuzzy ontology as the output.

3. 2. 3 Algorithm – II (FFCA – Merge Algorithm)

The FFCA-Merge algorithm for fuzzy ontology merging is described below:

i. It takes two source fuzzy ontologies and domain specific documents as input.

ii. Obtain fuzzy formal contexts of both fuzzy ontologies. If an instance of a concept exists in a document then count the number of instances of that concept and take ratio with the number of instances of all concepts of that fuzzy ontology in that document.

iii. Perform concept disambiguation and obtain a merged fuzzy formal context.

iv. Obtain fuzzy concept lattice from the merged fuzzy formal context. Here use TITANIC algorithm with modified support function.

v. Make fuzzy formal concept having empty set as key set as the root of the merged fuzzy ontology.

vi. Copy all fuzzy formal concepts having a single key set of cardinality one to the merged fuzzy ontology.

vii. Explore all fuzzy formal concepts having more than one key set of cardinality one and merge the corresponding source fuzzy ontology concepts.

viii. Explore all fuzzy formal concepts having a key set of cardinality two and establish the new relations.

ix. Copy the relations from the source fuzzy ontologies to the merged fuzzy ontology.

Pseudo code of Algorithm-II

In the Figure 3.8 we have given pseudo code of the algorithm that we presented above.

```
Input : Two fuzzy ontologies $FO_1$ and $FO_2$ and a set of Domain specific documents
Output: Merged Fuzzy Ontology MFO
Process:
      Begin
           Obtain Fuzzy Formal Contexts (FFC1 and FFC2) of $FO_1$ and $FO_2$;
           Perform Concept Disambiguation on FFC1 and FFC2;
           MFFC = Merge FFC1 and FFC2;
           Fuzz_Latt = Generate_FuzzyLattice(MFFC);
           MC = "";
           For all fuzzy concepts FC of Fuzz_Latt do
           Begin
                If (is_empty(Key_Set(FC))) then
                      Make FC root of the merged fuzzy ontology MFO;
                Else if(Single_Key_Set(FC)) then
                     If(Cardinality(Key_Set(FC)) == 1) then
                          Copy FC to merged fuzzy ontology MFO;
                     Else (Cardinality(Key_Set(FC)) == 2)
                          Establish relation between concepts of Key_Set(FC) in
                          merged fuzzy ontology MFO;
                Else
                     K_Set[] = Obtain_Key_Sets(FC);
                     For all i  do  Begin
                          If(Cardinality(K_Set[i]) == 1) then
                                 MC = Merge(MC, K_sey[i]);
                          Else (Cardinality(K_Set[i]) == 2)
                                 Establish relation between concepts of K_Set[i] in
                                 merged fuzzy ontology MFO;
                     End For all;
                     If (MC != "")  then Enter MC as MFO concept;
            End For all;
            For all concepts C of MFO
             Begin
                  For all participants c of C
                       For all subconcepts s of c do begin
                            $s'$ = Find MFO's concept having  s  as participant
                            If(is_related(C, $s'$) == 0) then
                                   copy the relation between s and c to the merged fuzzy
                                   ontology MFO as relation between $s'$ and C;
                            Else
                                   rel_weight ( $s'$ and C) = avg(rel_weight($s'$ and C),
                                                                      rel_weight( s, c));
                     End For all;
             End For all;
       End Process;
```

Figure 3.8. Pseudo code of the Algorithm - II

3. 2. 4 Implementation

In this section, we describe some important functions involved in the implementation of FFCA – Merge algorithm.

m_ontology

Main module generates merged fuzzy ontology concepts by exploring the fuzzy concepts of the fuzzy concept lattice. Then main module calls this function to copy relations from the source fuzzy ontologies to the merged fuzzy ontology. This function follows the procedure explained in the algorithm and copy the relations from the source fuzzy ontologies to the merged fuzzy ontology.

titanic_gen

This function generates a set of candidate sets. These sets are not the actual key sets. Main module prunes this set and generates actual key sets by using the pruning procedure given in the TITANIC algorithm. Closures of the key sets are found in the main module it self. These closures are nothing but intents of the fuzzy formal concepts of the lattice.

weight

Supports (weights) of the candidate sets and the key sets are used for the pruning process. This is the implementation of the support function. In addition to pruning, supports are also used for the closures computation and the candidate sets generation.

Chapter 4

Experiments and Results

In this chapter we give a detailed discussion about various experiments that are conducted to test the performance of SBFOM Algorithm and FFCA-Merge Algorithm developed in the previous chapter. For the SBFOM algorithm, which is based on the similarity measure, experiments have been conducted in an environment where agents providing various web services exchange their fuzzy ontologies and build local consensus fuzzy ontology in order to co-operate in providing services to users. For the FFCA-Merge algorithm, which is an extension of FCA–Merge to fuzzy ontology merging, experiments have been conducted on synthetic data.

The experiments are conducted on a Pentium 4, 1.80 GHz processor with 512 MB of RAM and running Microsoft Windows XP Operating System. C language is used for implementing both the algorithms. The ontologies that are used for conducting the experiments are taken from [Stephens and Huhns, 2001]. All these ontologies are developed by students for the People domain.

According to [Abulaish and Dey, 2006], a crisp ontology can be converted into a fuzzy ontology, where crisp relations are converted into fizzy relations by assigning weights as defined in Table 4.1. [Castano, et al, 2003] have divided relations into two categories linguistic relations and semantic relations, and assigned numeric weights to these relations. These weights are shown in Table 4.1. The weights associated with the terminological relationships are taken from ARTEMIS, where they have been tested on several real integration cases. The weights associated with semantic relations have been defined in HELIOS to express a measure of the strength of the concept connection posed by each relation for semantic similarity evaluation purposes. The higher is the weight associated with a semantic relation, the higher is the strength of the semantic connection between concepts. They have associated the weight 1.0 with properties since they are strongly related to a concept and provide its structural description.

	Relation	**Weight**
Linguistic interpretation	Synonym	1.0
	Hypernym/ Holonym	0.8
	Related term	0.5
Semantic interpretation	Property	1.0
	Same-as	1.0
	Kind-of	0.8
	Part-of	0.7
	Contains	0.5
	Associates	0.3

Table 4.1: Weights associated with terminological and semantic relations

4. 1 Evaluation of Similarity Based Fuzzy Ontology Merging Algorithm

Now we explain the working procedure of SBFOM algorithm with an example. Fuzzy ontologies shown in the Figures 4.1 and 4.2 are merged using this algorithm (with the similarity threshold 0.66). We refer these fuzzy ontologies as ***FO***$_1$ and ***FO***$_2$ respectively. Resulting merged fuzzy ontology is shown in the Figure 4.3. As a result of this merging process, the concept pairs from both fuzzy ontologies those got merged are ("Book", "Book"), ("Publication", "Publication"), ("Volume", Volume"), ("Journal", "Journal"), and ("Magazine", "Magazine"). Remaining steps of the algorithm are explained with the help of one merged concept that is ("Book", "Book"). "Book" concept of ***FO***$_1$ got merged with "Book" concept of ***FO***$_2$.

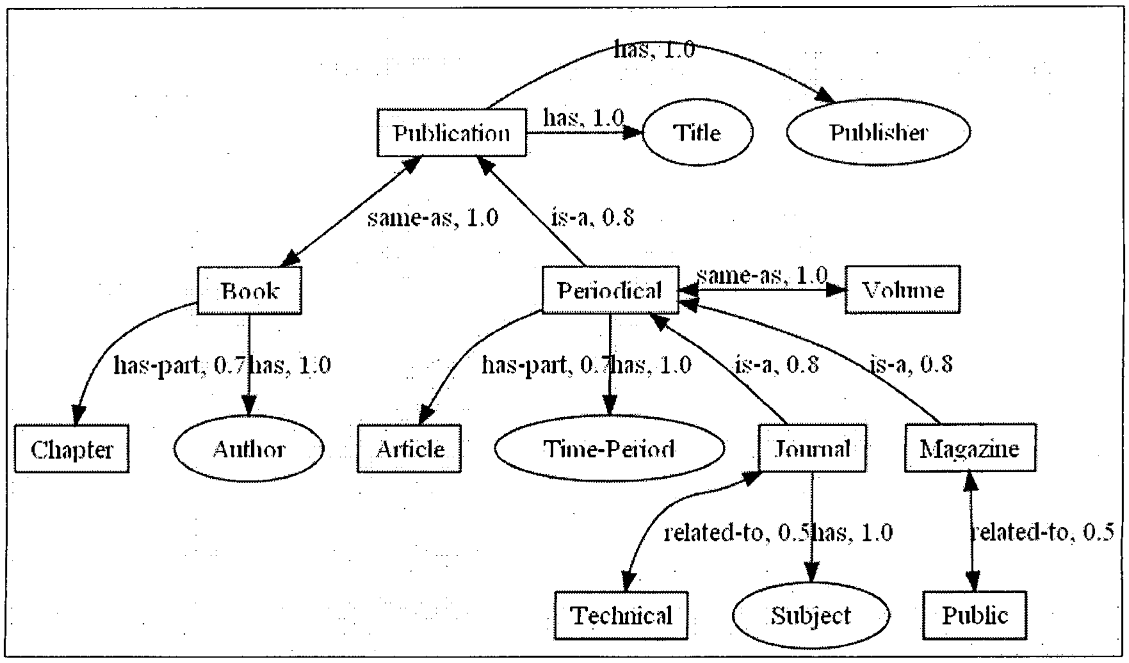


Figure 4.1. Fuzzy Ontology (***FO***$_1$) taken from [Abulaish and Dey, 2006]

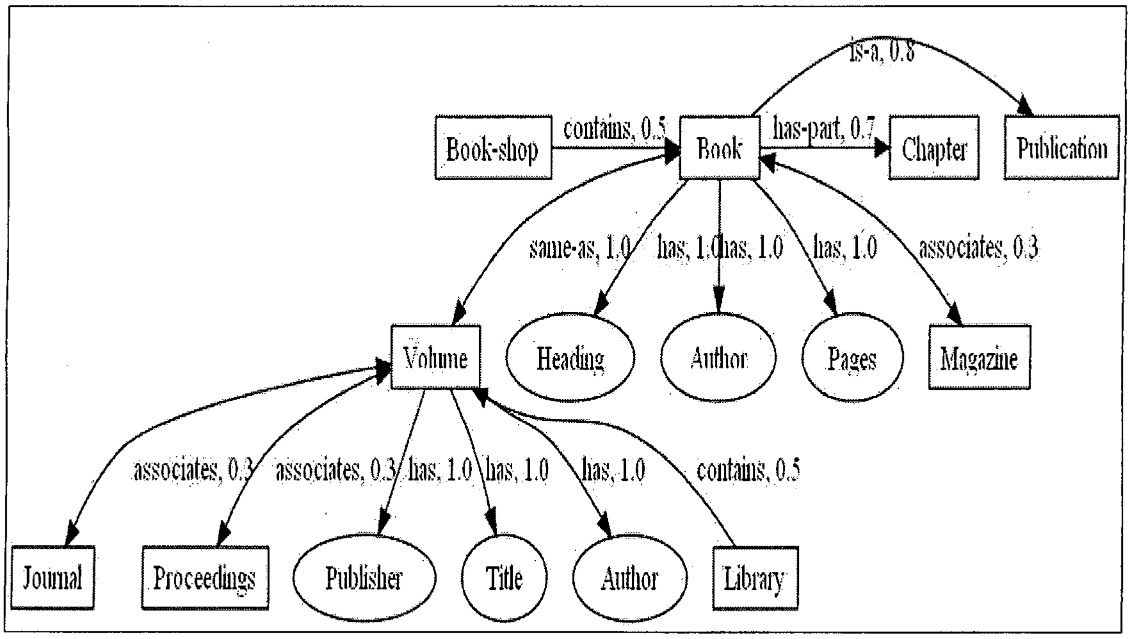


Figure 4.2. Fuzzy Ontology (***FO***$_2$) taken from [Abulaish and Dey, 2006]

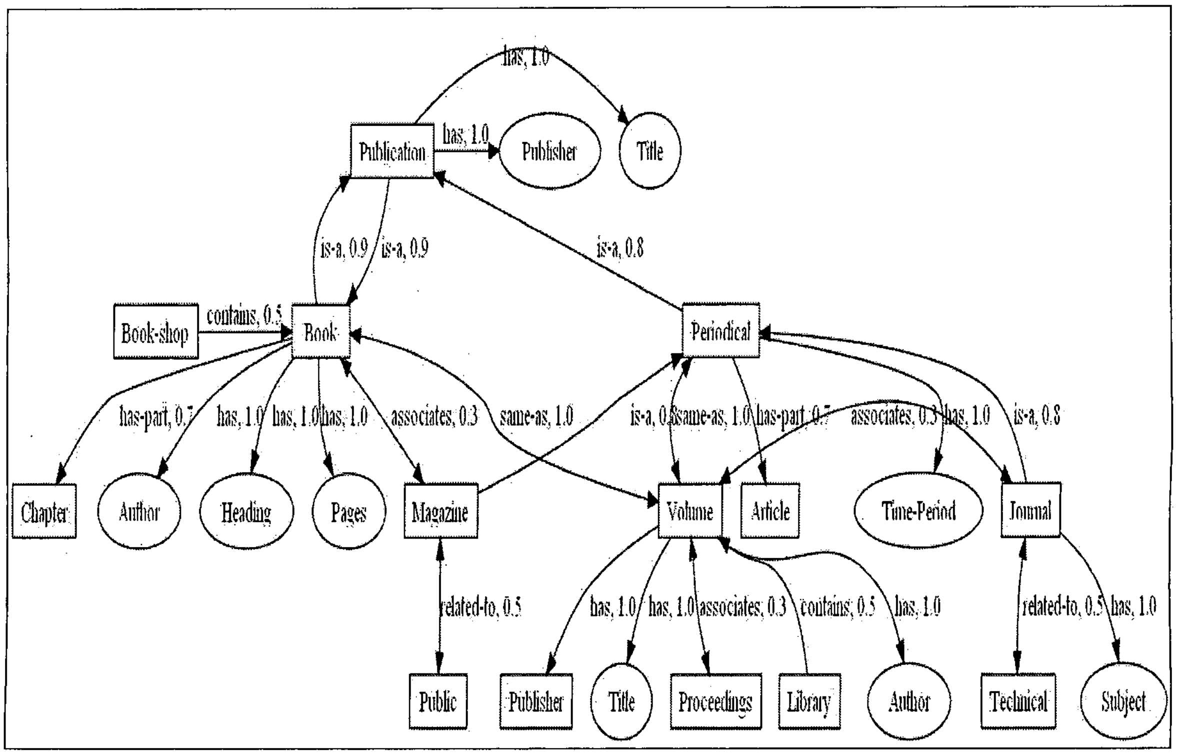


Figure 4.3. Merged Fuzzy Ontology of ***FO₁*** and ***FO₂***

And their super concepts named "Publication" from both fuzzy ontologies have also got merged. Membership values of fuzzy relations between these two concepts (Book, Publication) are 0.8 and 1.0 in ***FO_1*** and ***FO_2*** respectively. So the membership value of the relation between these two concepts in the merged fuzzy ontology is set to the average of the two membership values that is 0.9 according to this algorithm. According to the next step the non-merged concepts related to the "Book" concept in ***FO_2*** are copied to the merged fuzzy ontology. In the present example a non-merged concept related to "Book" is "Chapter", is copied to the merged fuzzy ontology as a result of this step. Then properties of "Book" concept from ***FO_2*** ("Heading", "Pages") are copied as properties of "Book" concept of the merged fuzzy ontology. This same process is repeated for all other merged concepts.

4. 1. 1 An Interesting Example

The crisp ontologies shown in the Figure 4.4 have been taken from the website http://www.daml.org/ontologies. These ontologies are structural representation of concepts related to academic organizations. Ontology management tool PROMPT which is a plug-in of Protégé, an ontology development environment, has been down loaded. Ontologies shown in the Figure 4.4 are merged using the PROMPT tool. While performing this merge tool's suggestions are followed and only suggested operations are performed. Above mentioned ontologies are also merged with SBFOM algorithm (with the similarity threshold 0.85) and the resultant merged ontology is compared with the result obtained from PROMPT. By observing these two results we have noticed two interesting points.

U Ontology

- Publication
- Academic_Mission
- Email
- Industrial_mission
- Office
- Organisation
    - Academic_org
    - Government_org
    - Industrial_org
- Person
    - Employee
- Project
- Research_Deliverables
- Research_Group
- Sex
- Work_Activity
- Employment_Categories
    - Director
    - Faculty
        - Teaching
        - Research
            - Postdoc
            - Project_Scientist
            - Students
    - Management
    - Staff

UMD Ontology

- Webresource
- Schedule
- Activity
- Address
- Agent
    - ArtificialAgent
    - SocialGroup
        - Organization
            - CommercialOrganization
            - EducationOrganization
            - GovernmentOrganization
            - NonprofitOrganization
        - Person
            - Student
                - GraduateStudent
                - UndergraduateStudent
            - Employee
- Event
- Location
- PhysicalObject
- Gender

Figure 4.4. Two ontologies taken from http://daml.org.ontologies

First, while merging ontologies using PROMPT tool, "Publication" concept of CMU ontology got merged with "Location" concept of UMD ontology. If the user who is performing merging is not an expert and follows suggestions of the tool, this type of results may be obtained. Where as in the result obtained after using SBFOM algorithm, this did not happen. A user who is not an expert of ontology design can also use this algorithm without generating the result mentioned above.

Second, SBFOM algorithm has also established four new equivalence relationships between Government_org and GovernmentOrganization, Industrial_org and CommercialOrganization, Academic_org and EducationOrganization, and Sex and Gender.

4. 1. 2 Building Local Consensus Fuzzy Ontology

We have evaluated the performance of this algorithm in an environment where agents are used for enterprise integration. According to [Williams, et al, 2005], an agent can incorporate ontological representations of its parent organization to broker Business-to-Business interoperability. Suppose an agent is searching for a particular web service. The ontological representations used by this agent for that web service may not match with the ontological representation used by the other agent which is capable of fulfilling its needs. To overcome this problem, these agents must be able to relate their ontological representations to each other in order to build local consensus ontology to find matches between the services they required and the services that other agents can provide. We listed below the goals behind conducting experiments in this environment.

- To test whether we are reaching local consensus while merging fuzzy ontologies of other agents or not,
- To test how the thesaurus built from WordNet affects the number of new relations learned.

Initially every agent is having its own perspective about a particular domain in the form of a fuzzy ontology. When an agent comes in contact with other agents they exchange their fuzzy ontologies and merge them to build a local consensus fuzzy ontology. Since we could not get fuzzy ontologies, ontologies obtained from the web site [Stephens and Huhns, 2001] are converted into fuzzy ontologies by assigning weights to relations as specified in the Table 4.1. In these experiments the value chosen for the similarity threshold is 0.75.

Merging Reaches Local Consensus

We are merging to one agent's fuzzy ontology, the fuzzy ontologies of all other agents to build a local consensus fuzzy ontology. Initially when we merge two or three fuzzy ontologies, this agent discovers and adds new concepts very rapidly. So the number of concepts in the consensus fuzzy ontology increases rapidly in the beginning. Later as many fuzzy ontologies are getting merged, the number of concepts in the consensus fuzzy ontology increases slowly, because most of the concepts in other agent's fuzzy ontology are already understood by this agent.

This fact is demonstrated in our experiments and the results of these experiments are shown in the Figure 4.5. The Graph in the Figure 4.5 shows, how the number of concepts in the consensus fuzzy ontologies changes when fuzzy ontologies are getting merged with consensus fuzzy ontology. The facts; a rapid increase initially in the number of concepts in the local consensus fuzzy ontology, and a slow increase later in the number of concepts in the local consensus fuzzy ontology, have been observed in this graph. This observation proves that we reach the local consensus in this domain.

While building local consensus fuzzy ontology, in the initial phases of the merge process the percentage of concepts found similar is very less, because most of the concepts in other agent's fuzzy ontology are not known to this agent. Later in the process as many fuzzy ontologies are getting merged to this agent, the percentage of similar concepts increases, because most of the concepts in other agent's fuzzy ontology are known to this agent. This is a symbol of reaching local consensus fuzzy ontology. This fact is observed in our experiments. In the Figure 4.6 we can see that, as more fuzzy ontologies are getting merged the percentage of similar concepts increases.

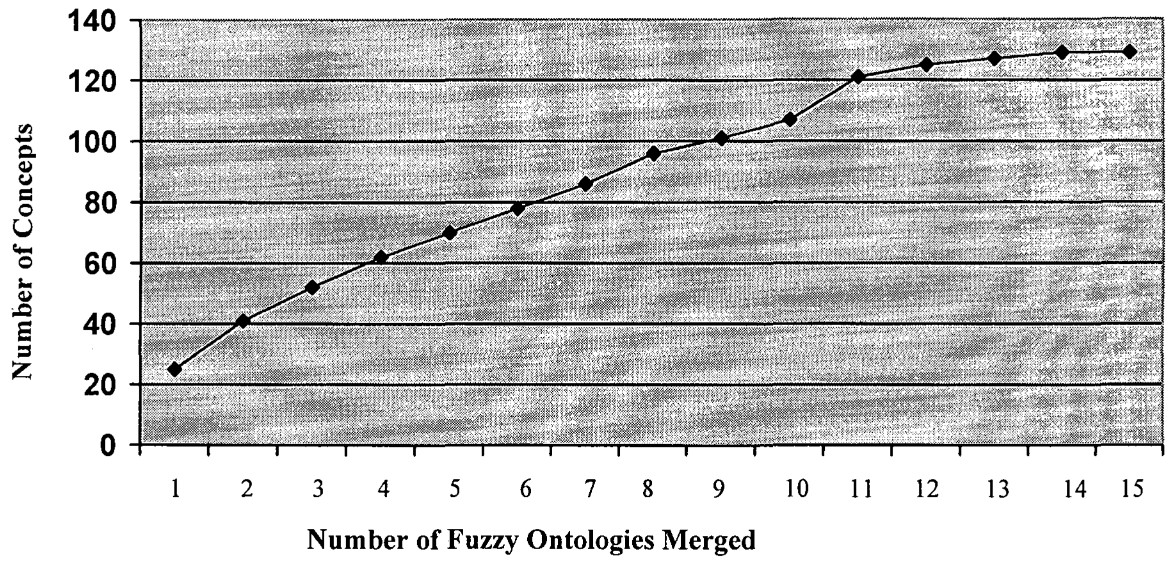


Figure 4.5. Graph showing an increase in the number of concepts in the local consensus fuzzy ontology as the number of fuzzy ontologies merged increases

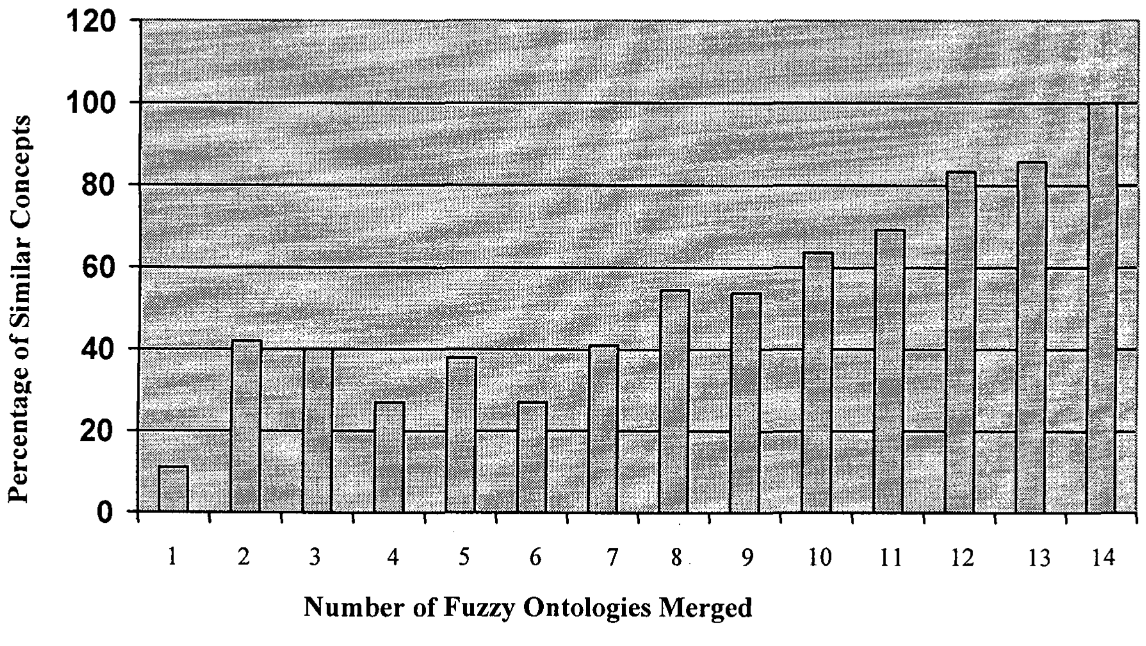
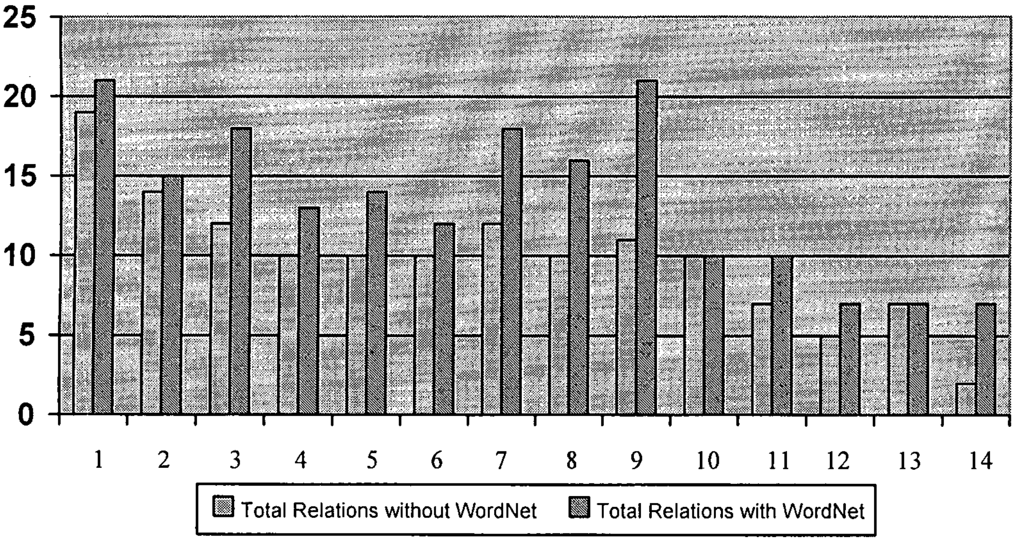


Figure 4.6. Graph showing an increase in the percentage of similar concepts as the number of fuzzy ontologies merged increases

New Relations Learned and Discovered

As we explained in previous sections, "relations discovered" (relations known to other agent and not known to this agent) is different from "relations learned" (new relations learned between concepts of other agent and concepts of this agent (consensus fuzzy ontology) with the help of WordNet). The number of new relations discovered depends on the number of new concepts discovered. If the discovered concepts are more (or less) then the discovered relations are also more (or less).



Number of Fuzzy Ontologies Merged

Figure 4.7. Graph showing the number of new relations added with and withoout WordNet as the number of fuzzy ontologies merged increases

The number of new relations learned depends on how the discovered concepts are related to the concepts already known to this agent (consensus fuzzy ontology). If these discovered concepts are related to more concepts of this agent, then the number of relations learned is more. If these discovered concepts are related to less number of concepts of this agent then the number of new relations learned is going to be low.

In the Figure 4.7 we can see that, when using WordNet the number of new relations added to the local consensus fuzzy ontology is more as compared to the number of new relations added when not using WordNet. When not using WordNet new relations added to the consensus fuzzy ontology are only discovered relations, that is, relations known to other agent and not known to this agent. With the help of WordNet new relations added to the consensus fuzzy ontology are both discovered and learned relations. From the Figure 4.7 we can observe that by using WordNet we can learn more relations among concepts.

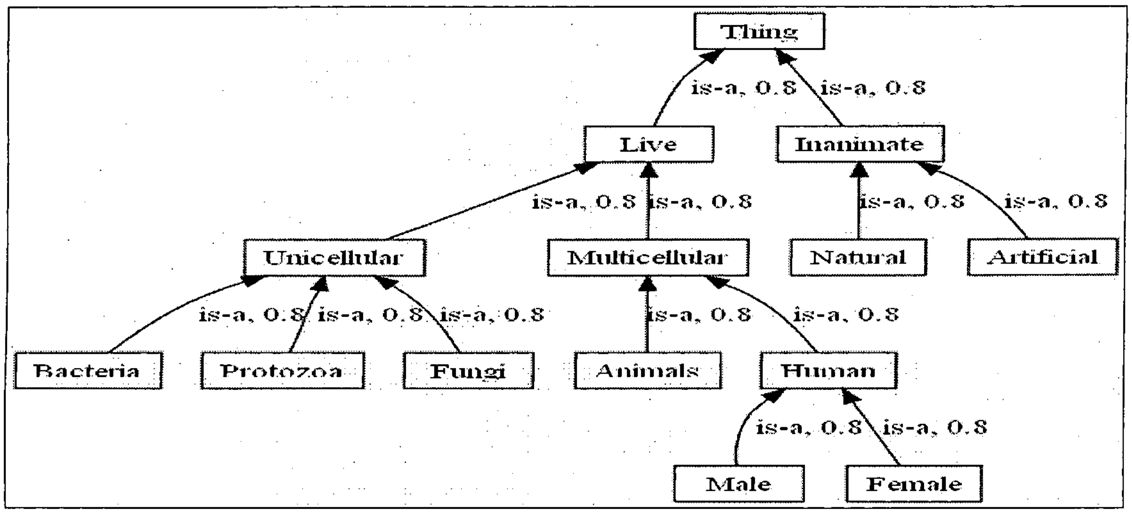


Figure 4.8. Fuzzy Ontology of agent 1

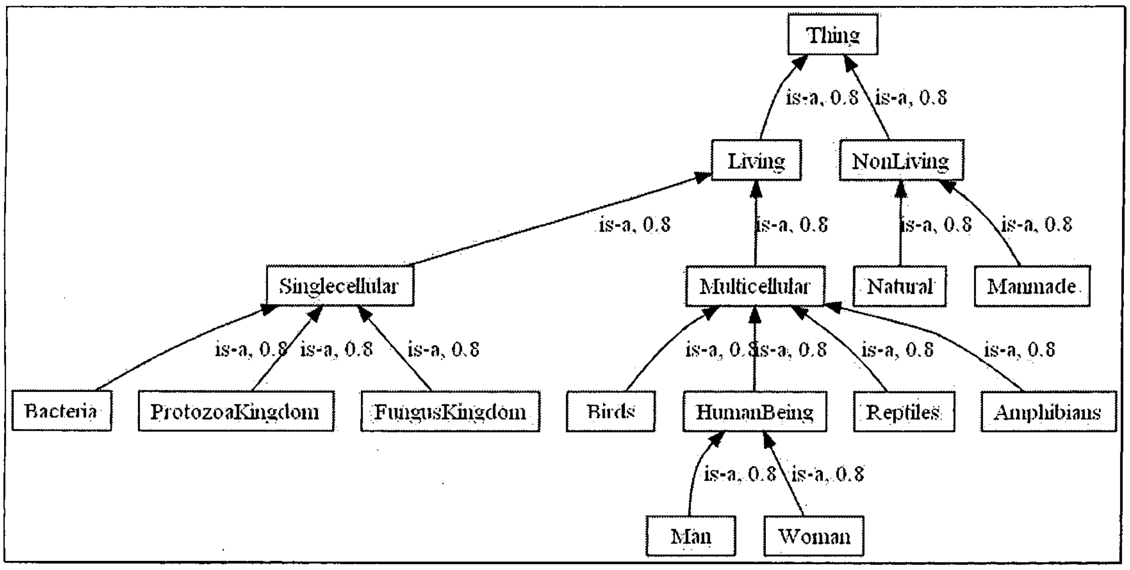


Figure 4.9. Fuzzy Ontology of agent 2

Even though by not using the thesaurus (WordNet) better performance in terms of speed can be obtained. But some semantic matches we have to loose. To explain this we have merged two fuzzy ontologies belonging to two agents with and without using the thesaurus.

We have merged the fuzzy ontologies that are shown in the Figures 4.8 and 4.9 without using WordNet. The resultant fuzzy ontology is shown in the Figure 4.10. In these figures (Figure 4.8 and Figure 4.9), concepts named “Unicellular” and “Single cellular” are semantically similar. But since machine is unaware of this fact it cannot establish any relation between these two concepts. To provide this semantic knowledge we use thesaurus or lexical database built from the WordNet. The result of merging both fuzzy ontologies that are shown in the Figures 4.8 and 4.9 with using thesaurus is shown in the Figure 4.11, in which semantic equivalence relation between above mentioned concepts is introduced and many more semantic equivalences are found.

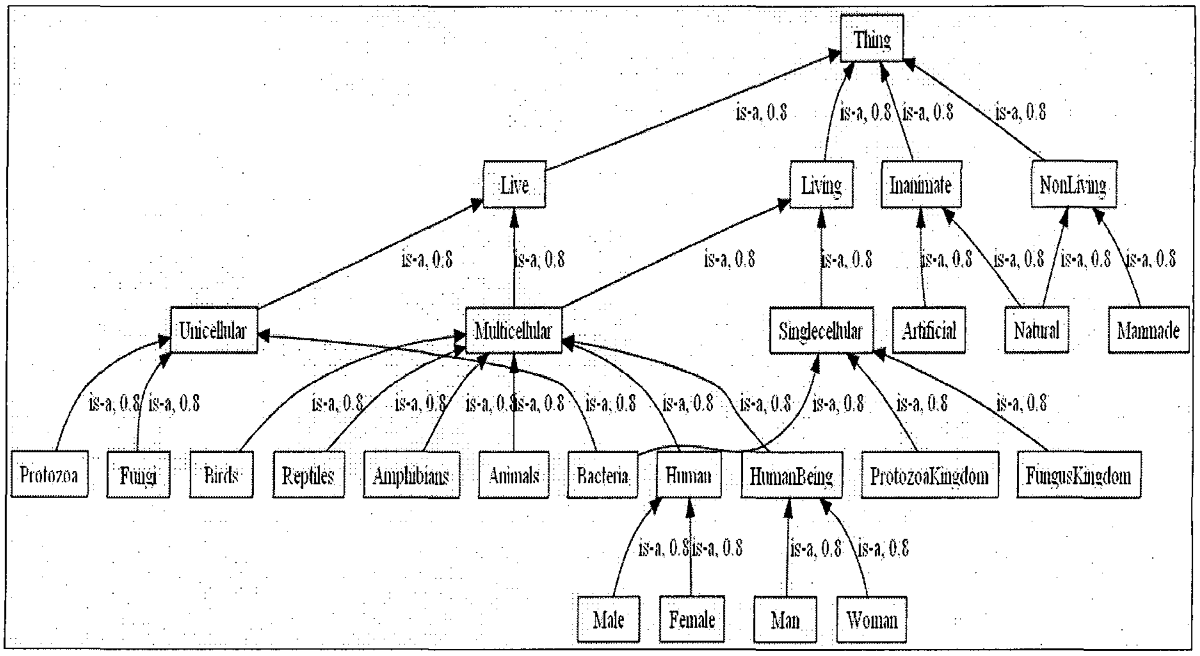


Figure 4.10. Merged fuzzy ontology of agent 1 and agent 2 without using WordNet

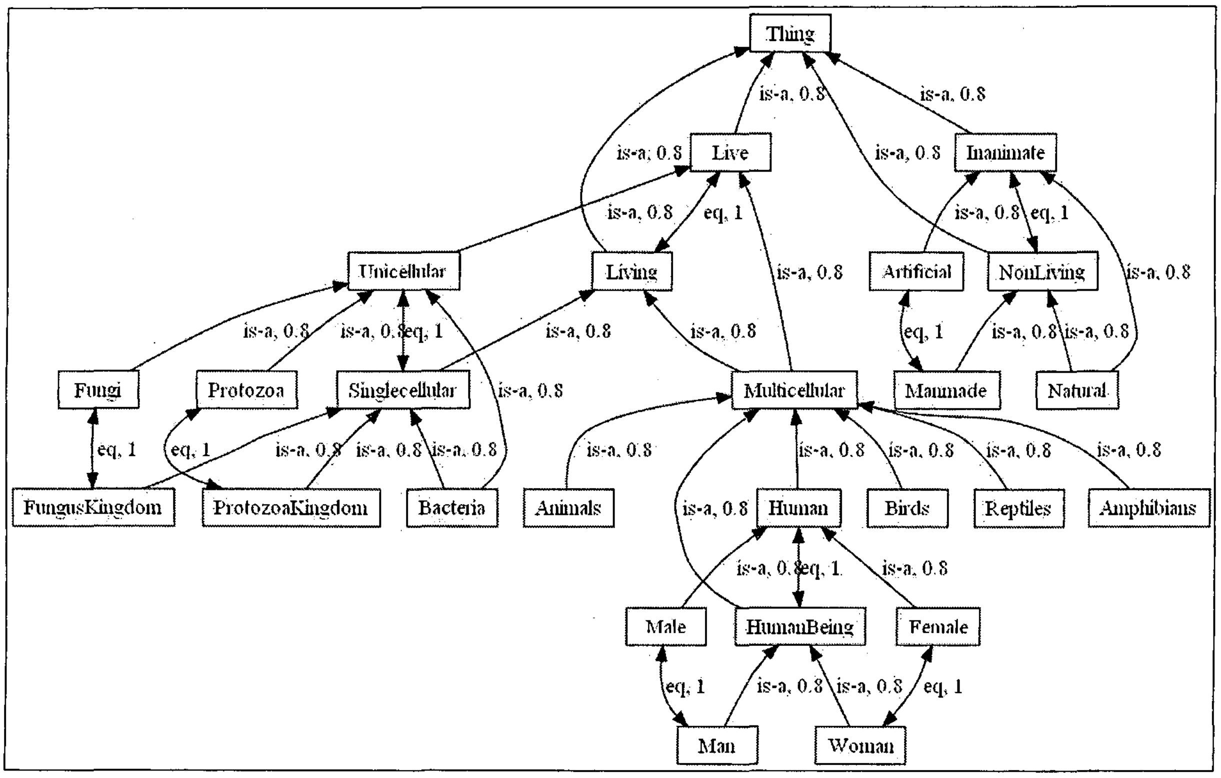


Figure 4.11. Merged fuzzy ontology of agent 1 and agent 2 using WordNet

4. 2 Evaluation of FFCA – Merge Algorithm

This algorithm has been tested on a data set that has been generated artificially since we could not get data to test this method. Fuzzy ontologies shown in the Figure 4.12 are inputs to our algorithm. Fuzzy formal contexts of them are shown in the Tables 4.2 and 4.3 with α = 0.5. Fuzzy formal concept lattice has been obtained using modified TITANIC algorithm after performing concept disambiguation, is shown in the Figure 4.13.

When we have generated merged fuzzy ontology from the fuzzy concept lattice, fuzzy concept labeled with {Thing_1, Thing_2} has become the root of the merged fuzzy ontology. Fuzzy concepts labeled with {Creature_2}, {Animal_1}, and {Socialised_1} are copied to the merged fuzzy ontology according to the algorithm. Fuzzy concepts labeled with {LivingThing_1, Animate_Being_2}, {Person_1, Human_2}, and

{Unsocialised_1, NonHuman_2} have suggested to merge those concepts from source fuzzy ontologies. Those concepts are merged and added to the merged fuzzy ontology.

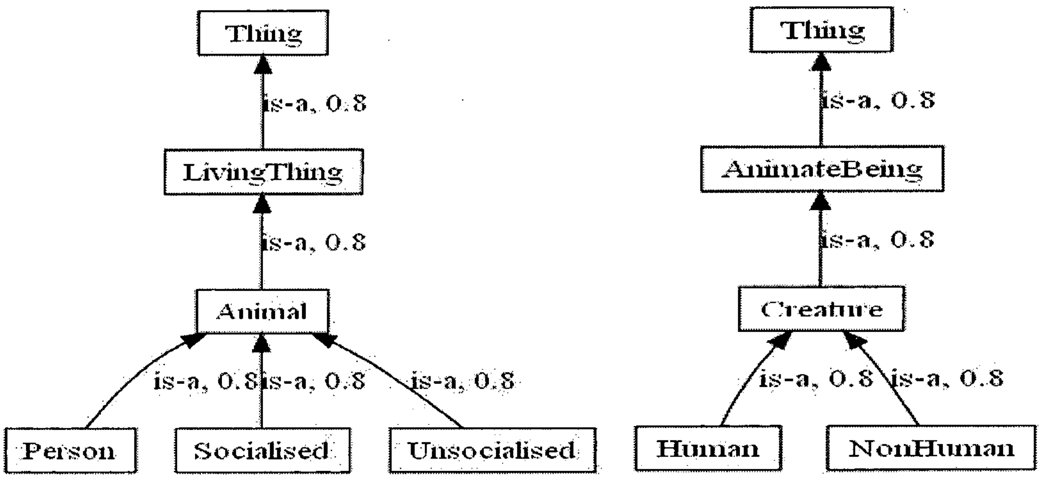


Figure 4.12. Fuzzy ontologies ***FO₁*** and ***FO₂***

The fuzzy concept that is above the concept labeled with the key sets {Person_1} and {Human_2} suggests to add a new relation between "Animal_1" and "Creature_2". Hence this new relation has been added the merged fuzzy ontology.

After, relations from the source fuzzy ontologies have been added to the merged fuzzy ontology. We started with the root labeled with concepts {Thing_1, Thing_2}. "Thing_1" is the first concept of the merged root. "LivingThing" is its subconcept. "LivingThing" is present in the merged concept {LivingThing_1, AnimateBeing_2}. So the relation between "Thing" and "LivingThing" from the first fuzzy ontology has copied to the merged fuzzy ontology as a relation between merged concepts {Thing_1, Thing_2} and {LivingThing_1, AnimateBeing_2} along with the membership value. "Thing_2" is the second concept of the merged root. "AnimateBeing" is its subconcept. "AnimateBeing" is present in the merged concept {LivingThing_1, AnimateBeing_2}. Since the relation has already been established between the corresponding merged concepts ({Thing_1, Thing_2} and {LivingThing_1, AnimateBeing_2}), according to our algorithm membership value of this relation has been modified as the average of the existing membership value of this relation and the membership value of the relation between "Thing" and "AnimateBeing". Similarly other relations have also been copied. The resultant merged fuzzy ontology is shown in the Figure 4.14, in which the concept names of the first fuzzy ontology are used as representatives of the merged concepts.

	Thing	**Living Thing**	**Animal**	**Person**	**Socialised**	**Unsocialised**
D1	1.0	0.8	0.9	0.8	0.8	0.8
D2	1.0	0.8	0.9	0.7		0.6
D3	1.0	0.9	0.7			
D4	1.0	0.7	0.9	0.7	0.6	0.75
D5	1.0	0.8				
D6	1.0	0.9	0.8			
D7	1.0	0.8	0.7			
D8	1.0	0.7	0.8	0.7	0.75	0.9
D9	1.0	0.8	0.9	0.8	0.7	0.8
D10	1.0	0.9	0.8	0.7	0.6	0.7
D11	1.0	0.7	0.9	0.7		0.6
D12	1.0	0.8	0.8			
D13	1.0	0.9	0.7			
D14	1.0	0.7	0.8	0.9	0.9	

Table 4.2: Fuzzy Formal context of $\boldsymbol{FO_1}$

	Thing	**Animate Being**	**Creature**	**Human**	**NonHuman**
D1	1.0	0.8	0.7	0.8	0.8
D2	1.0	0.8	0.8	0.7	0.6
D3	1.0	0.9	0.9		
D4	1.0	0.7	0.8	0.7	0.75
D5	1.0	0.8	0.7		
D6	1.0	0.9	0.9		
D7	1.0	0.8			
D8	1.0	0.7	0.9	0.7	0.9
D9	1.0	0.8	0.7	0.8	0.8
D10	1.0	0.9	0.7	0.7	0.7
D11	1.0	0.7	0.8	0.7	0.6
D12	1.0	0.8			
D13	1.0	0.9	0.9		
D14	1.0	0.7	0.75	0.9	

Table 4.3: Fuzzy Formal context of $\boldsymbol{FO_2}$

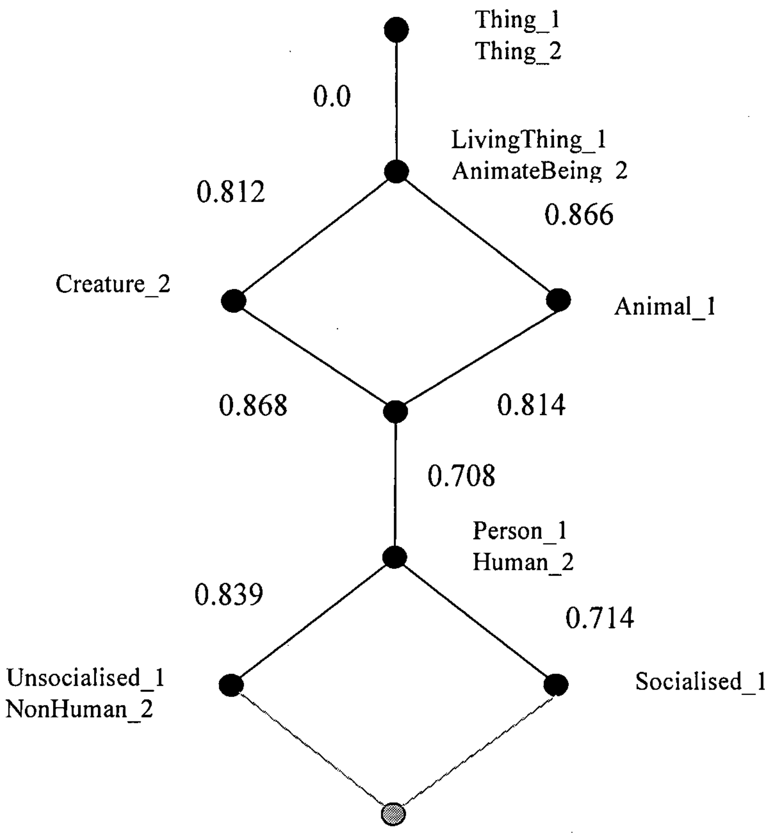


Figure 4.13. Fuzzy concept lattice of $\boldsymbol{FO_1}$ and $\boldsymbol{FO_2}$

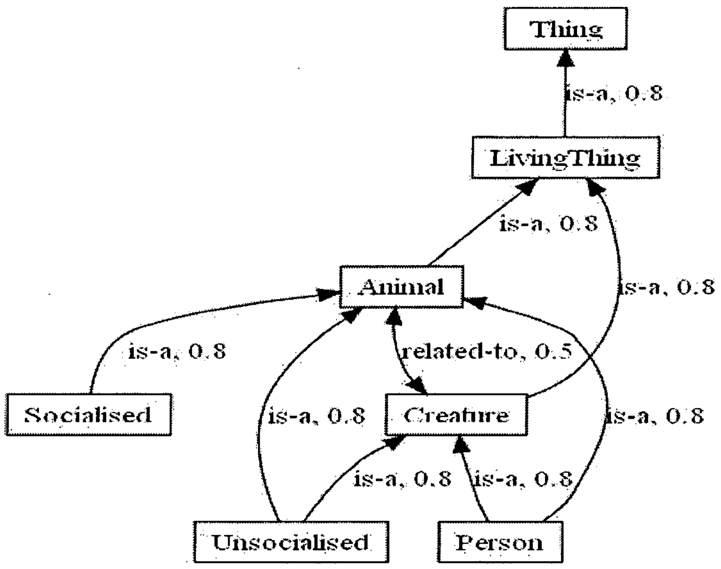


Figure 4.14. Merged fuzzy ontology of $\boldsymbol{FO_1}$ and $\boldsymbol{FO_2}$

Chapter 5

Conclusions

In this thesis we developed two fuzzy ontology merging algorithms: Similarity based fuzzy ontology merging algorithm and Fuzzy FCA – Merge algorithm. Similarity based fuzzy ontology merging algorithm uses the weighted sum of lexical similarity, linguistic similarity and contextual similarity as a similarity measure to find matching concepts. Through the experiments we have demonstrated the performance of this algorithm in building local consensus fuzzy ontology. While comparing the results of this algorithm with that of PROMT tool for two input fuzzy ontologies, it is observed that the SBFOM algorithm produces results similar to PROMPT tool. Also this algorithm generates new equivalence relations between concepts of two input fuzzy ontologies.

In FFCA – Merge algorithm we have extended FCA – Merge to the fuzzy ontology merging by obtaining fuzzy formal contexts instead of crisp formal contexts of given input fuzzy ontologies, by constructing the fuzzy concept lattice instead of crisp concept lattice, and by developing the procedure for copying relations from the source fuzzy ontologies to the merged fuzzy ontology. We tested this algorithm on synthetic data and the merges and relations produced illustrate its performance. In future, one can study how to obtain weights of the suggested relations from the fuzzy concept lattice itself. Further work would be required to develop tools for merging fuzzy ontologies based on these proposed algorithms.

References

Abulaish, M. and Dey, L. (2006), "Interoperability among Distributed Overlapping Ontologies – A Fuzzy Ontology Framework", *In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI '06), Hong Kong, pp. 397-403.*

Berners-Lee, T., Hendler, J., and Lassila, O. (2001), "The Semantic Web", May, Scientific American, Feature Article.

Calgeri, S., and Sanchez, E. (2007), "A Fuzzy Ontology – Approach to Improve Semantic Information Retrieval", *URSW 2007, Busan, Korea, pp. 125-130.*

Castano, S., Ferrara, A., and Montanelli, S. (2003), "H-Match: An Algorithm for Dynamically Matching Ontologies in Peer-based Systems", *In Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB '03), Berlin, Germany, pp. 231-250.*

Chalupsky, H. (2000), "OntoMorph: A Translation System for Symbolic Knowledge", *In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR 2000), Breckenridge, Colorado, USA.*

Ganter, B., and Wille, R. (1999), *"Formal Concept Analysis: Mathematical Foundations*", Springer.

Graphviz (2004), A Graph Visualization Software, http://www.graphviz.org.

Gruber, T., R. (1993), "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", *Int. J. of Human-Computer Studies, Vol. 43, Issue 5-6, pp. 907-928.*

Gruber, T., R., and Olsen, G., R. (1994), "An Ontology for Engineering Mathematics", *In Jon Doyle, Piero Torasso, and Erik Sandewall, Eds., Fourth International Conference on Principles of Knowledge Representation and Reasoning, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann.*

Guarino, N. (1995), "Formal Ontology, Conceptual Analysis and Knowledge Representation", *International Journal of Human-Computer Studies, Vol. 43, No. 5-6, pp. 625-640.*

Heflin, J. and Hendler, J. (2001), "A Portrait of the Semantic Web in Action", IEEE Intelligent Systems, Vol. 16, Issue. 2, pp. 54-59.

Lee, Chang-Shing, Jian, Zhi-Wei, and Huang, Liu-Hai. (2005), "A Fuzzy Ontology and Its Application to News Summarization", *IEEE Transactions on Systems, Man, and Cybernetics, Part-B, Vol. 35, No. 5, pp. 859-880.*

McGuinness, Deborah L., Fikes, R., Rice, J., and Wilder, S. (2000), "An Environment for Merging and Testing Large Ontologies", *In Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR 2000) , Breckenridge, Colorado, USA, pp. 12-15.*

Miller, G. (1995), "WordNet: a lexical database for English", *Communications of the ACM, Vol. 38, No. 11, pp. 39-41.*

Mitra, P., Wiederhold, G., and Kersten, M. (2000), "A Graph-Oriented Model for Articulation of Ontology Interdependencies", *EDBT 2000, Konstanz, Germany, Springer, LNCS 1777, pp. 86-100.*

Noy, N., F., and Musen, M., A. (1999), "SMART: Automated Support for Ontology Merging and Alignment", *In Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, LNCS, Vol. 1937.*

Noy, N., F., and Musen, M., A. (2003), "The PROMPT Suite: interactive tools for ontology merging and mapping", *International Journal of Human-Computer Studies, Elsevier, Vol. 59, pp. 983-1024.*

Parry, D. (2004a), "A fuzzy ontology for medical document retrieval", *In Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation, ACM International Conference Proceeding Series, Vol. 32, pp. 121-126.*

Parry, D. (2004b), "Fuzzification of a standard ontology to encourage reuse", *In Proceedings of the 2004 IEEE International conference on Information Reuse and Integration, Las Vegas, NV, USA, IEEE Systems, Man, Cybernetics Society, pp. 582-587.*

Sanchez, E., and Yamanoi, T. (2006), "Fuzzy ontologies for the semantic web", Springer, *LNAI 4027, pp. 691-699.*

Shah, U., Finin, T., Joshi, A. Cost, R., S., and Mayfield, J. (2002), "Information retrieval on the semantic web", *In proceedings of Conference on Information and Knowledge Management, McLean, Virginia, USA, CIKM '02, ACM, pp. 461-468.*

Stephens, L., M., and Huhns, M., N. (2001), "Consensus Ontologies Reconciling the Semantics of Web Pages and Agents", *Internet Computing, IEEE, Vol. 5, No. 5, pp. 92-95.*

Stumme, G., Taouil, R., Bastide, Y., Passquier, N., and Lakhal, L. (2000), "Fast Computation of Concept Lattices Using Data Mining Techniques", *In Proceedings of 7th International Workshop on Knowledge Representation meets Databases, Berlin Germany, pp. 129-139.*

Stumme, G., and Maedche, A. (2001), "FCA-Merge: Bottom-Up Merging of Ontologies", *In Proceedings of the 7th International Conference on Artificial Intelligence (IJCAI '01), Seattle, WA, pp. 225-230.*

Stumme, G., Hotho, A., and Berendt, B. (2006), "Semantic Web Mining - State of the Art and Future Directions", *Journal of Web Semantics, Elsevier, Vol. 4, No. 2, pp. 124-143.*

Tho, Q., T., Hui, S., C., Fong, A., C., M., and Cao, T., H. (2006), "Automatic Fuzzy Ontology Generation for the Semantic Web", *IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 6, pp. 842- 856.*

Widyantoro, D., H., and Yen, J. (2001), "A fuzzy ontology–based abstract search engine and its user studies", *In Proceedings of the 10th IEEE International Conference on Fuzzy Systems, Melbourne, Australia, pp. 1291-1294.*

Williams, A., B., Padmanabhan, A., and Blake, M., B. (2005), "Experimentation with Local Consensus Ontologies with Implications for Automated Service Composition", *IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 7, pp. 969-981.*

Zadeh, L., A. (1965), "Fuzzy Sets", *Information and Control, Vol. 8, pp. 338-353.*

Zhou, W., Liu, Z., T., and Zhao, Y. (2007), "Ontology Learning by Clustering Based on Fuzzy Formal Concept Analysis", *31st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2007), Beijing, China, pp. 204-210.*