Université de Tunis
Institut Supérieur de Gestion de Tunis

Laboratoire de recherche LARODEC

**Mémoire présenté en vue de l'obtention du**
**Diplôme de Mastère en Informatique Appliquée à la Gestion**

# BeliefOWL
# An Evidential Extension to OWL

**Elaboré par:**
**Amira Essaid**

**Dirigé par:**
**Dr. Boutheina Ben Yaghlane Ben Slimen**

Octobre 2008

# Acknowledgments

Numerous people have contributed to this dissertation in various ways. I am deeply indebted to all of them for their support and would like to offer my thanks.

First, I am deeply grateful to my supervisor Dr Boutheina Ben Yaghlane for her academic guidance and her strong support. She was always there when I needed advice and provided me her recommendations to improve this work. Many thanks to her for her assistance and her vital encouragement.

I send my sincerest thanks to my parents and my two sisters for their love, their moral support and their understanding. They tried to create for me a pleasant environment to achieve this master.

Finally, I wish to thank my friends who encouraged me and motivated me.

# Abstract

In semantic web environment, ontologies have become used to represent a domain of discourse to facilitate knowledge sharing and information exchange. Many languages have been developed to represent ontologies, among them the web ontology language (OWL) which cannot capture the uncertainty about the concepts in a domain. To address the problem of modeling uncertainty in semantic web, we propose in this master thesis, a tool called BeliefOWL based on an evidential approach. It focuses on translating an ontology into an evidential network by applying a set of structural translation rules and then belief masses will be assigned to the different nodes in order to propagate uncertainties. The originality of our contribution consists in applying the Demspter-Shafer theory in two tasks: the ontology representation and reasoning.

# Résumé

Dans le web sémantique, les ontologies sont utlisées pour représenter un domaine afin de faciliter le partage des connaissances et l'échange de l'information. Beaucoup de langages ont été développés pour représenter ces ontologies parmi lesquels on cite le langage OWL qui est incapable de tenir compte de l'incertitude. Dans le but de traiter le problème de modélisation de cette incertitude dans le web sémantique, nous proposons dans ce mémoire un outil nommé BeliefOWL qui est basé sur la théorie de l'évidence et qui cherche à transformer une ontologie en un réseau évidentiel en appliquant un ensemble de règles structurelles, par la suite des masses seront attribuées aux différents noeuds du réseau afin de permettre de réaliser une propagation. L'originalité de notre contribution réside dans l'application de la théorie de l'évidence dans deux tâches à savoir la représentation d'une ontologie et le raisonnement.

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

## 1.1 Motivation and Purpose

With the emergence of the semantic web, the vision to the web have changed. In fact it becomes not only a way for interchanging documents but also a way for making the information understandable by machines. The interchange of this information is not so easy. The data is located in distributed and heterogeneous systems so the focus today is how to assure an information sharing and the integration between systems.

To make the information sharing and the communication between machines possible and efficient, two main problems that can occur must be solved: the information retrieval and the interoperability (Wache et al., 2001). The former one consists in finding the most suitable source containing the information needed whereas the latter can be viewed as bringing together heterogeneous and distributed computer systems and make the data well interpreted. In order to achieve semantic interoperability in an heterogeneous information system, the meaning of the information interchanged has to be understood across the systems. Ontologies are required as the most important component in helping to achieve the semantic integration interoperability and reconciliation.

Ontologies are structured in a formal vocabulary of terms describing the concepts related to a specific domain. Moreover an ontology specifies the relations existing between these concepts. This taxonomy helps the applications to explore the content of the information shared. To represent this structure of concepts and make it understood by machines, many languages have been developed among them the web ontology language OWL (Antoniou and Van Harmelen, 2004). In addition to the static structure of an ontology which is one of the first concerns of the ontology research, the ontology reasoning is another important task that should be carried across an ontology enabling systems to infer implicit knowledge.

OWL, developed as a vocabulary extension of the RDF (Resource Description Framework), is a semantic markup language for publishing ontologies on the World Wide Web. The OWL is based on crisp logic and is unable to handle incomplete or partial information about the concepts or the properties related to a domain. Taking into account the uncertainty aspect in an ontology representation and even in reasoning seems to be an important problem to address. Many researchers applied uncertainty

theories (probability theory, fuzzy sets, possibility theory...) to represent the ontology and reason across it. Some of them have chosen to translate an ontology into a Bayesian network as a formalism to capture the uncertain information in order to enable the inference later (Ding, 2005), (Yang and Calmet, 2005). This choice of Bayesian networks is justified by the fact of structural similarity existing between an ontology and a bayesian network as well as the probabilistic reasoning capability of this kind of networks.

Most of the works are based on probability theory but the introduction of the Dempster-Shafer theory and considering its advantages encourage some researchers to apply it in the ontology research tasks such as in ontology mapping (Nagy et al., 2006), (Laamari and Ben Yaghlane, 2007) but few are those who use it for ontology representation and reasoning. At this stage, we are interested to use this theory and especially we are encourage to work with the evidential networks which are viewed as effective and more appropriate graphical representation for uncertain knowledge.

The main purpose of this master is to study the ontology representation approach based on the evidential theory. This dissertation addresses the extension of OWL ontology with belief functions as well as the translation of this ontology into an evidential network in order to propagate beliefs later. It describes the design of our tool BeliefOWL as an approach to handle the uncertainty inherent to the main ontology research tasks.

## 1.2 Master Contributions

Once we explore the background material of the ontology representation and reasoning based on the probability theory and fuzzy sets theory, we will define the theoretical aspect of our approach by addressing the issues that encourage us to propose our BeliefOWL tool. This tool is able to extend an ontology with belief information and to translate an ontology into an evidential network by applying a set of structural translation rules. The different steps leading to our tool will be described in detail and an algorithm will be presented as well as an illustrative application will be given.

## 1.3 Master Organization

In Chapter 2 we provide a brief introduction to the ontology research area in particular the ontology representation, the ontology reasoning and the ontology mapping in order to get a general idea of this area and have the knowledge of the basics of these tasks. Our work is based on the ontology in particular those written with the OWL. For that purpose a detailed description of this language will be given.

Chapter 3 is devoted to present the uncertainty field. It focuses especially on the presentation of the Bayesian network as a graphical model used in the semantic web to facilitate the reasoning tasks later. In addition to that, we present in this chapter the works dealing with uncertainty in the ontology representation and reasoning and using

probability and fuzzy sets as an uncertainty approaches.

We present in Chapter 4 the motivations that incite us to propose our tool. We also describe the different steps leading to the tool, how we include the uncertainty in ontology representation and ontology reasoning. We then propose our architecture and our algorithm in addition to an example illustrating it.

Finally, Chapter 5 gives a summary of the results obtained in this dissertation and suggests what direction future research could go.

# 2
# Web Ontology Preliminaries

## 2.1 Introduction

Day after day the field of ontology research is knowing changes. It is a dynamic domain that grows and matures because the users' needs are changing, the questions for which the researchers are trying to find an answer to shifts as well. The issues and the first challenges in the ontology research concerned at the beginning the theoretical aspect but nowadays the focus is more oriented to the real-world and how the ontologies have to be used in large-scale applications.

In (Noy and Klein, 2004), the authors traced the evolution of ontology research by pointing that the first concern in this area was devoted to define the term *"Ontology"* and to specify the requirements that an ontology must satisfy. Later, the research focused on developing *representation languages* for the ontologies in order to represent them and to enable the applications to explore the content of the information. Having a static structure of an ontology represented by a specific language is not enough. The need to do *reasoning* tasks across the ontology to infer implicit knowledge seems to be a crucial interest in the ontology research and is one of the issue addressed in this area. At this stage, the main focus is on how to manipulate a single ontology by representing it formally and assuring reasoning tasks but with the appearance of a large number of ontologies, the interchange and the communication between them is essential, for that purpose many tools such as Glue (Doan et al., 2004), QOM (Ehrig and Staab, 2004), Omen (Mitra et al., 2005)... have been developed to assure the alignment between the ontologies which becomes one of the challenge addressed by the ontology research.

The aim of this chapter is to trace the evolution of the ontology research field by defining the ontology concept as well as presenting the different ontology modeling languages. A special concern will be devoted to the introduction of the Web Ontology Language (OWL) which is considered as the most descriptive one and selected as the underlying formal ontology language for our work. Inferring implicit knowledge and improving systems by making them able to do reasoning tasks is one of the topic addressed in this chapter. We will focus on the importance of the reasoning and the purposes of handling it as well as the different reasoning tasks that can be held across an ontology. In addition to these two ontology research's axes, a brief introduction to the ontology

mapping is given.

In this chapter, we will focus on the different steps of the evolution of the ontology research. In section 2.2 we will introduce the notion of semantic web and we will give the different definitions of the ontology concept as exposed in the literature, then in section 2.3 the different languages used to represent ontologies will be presented by concentrating especially on the OWL. Section 2.4 is devoted to expose the different reasoning tasks that can be done across an ontology. Finally we will give a brief presentation of ontology mapping in section 2.5.

## 2.2 Semantic Web and Ontology

### 2.2.1 Semantic Web

The current World Wide Web (WWW) is a syntactic web where the web pages are designed to be read by humans and not by computer programs which search for the information in response to users' queries without processing the semantics of the information manipulated. Although the WWW helped in assuring information exchange between applications but it still cannot assure the interoperation without some pre-existing agreements. To solve this problem, the new generation of the web, the Semantic Web, will add semantic information to the web resources in order to make their content understandable by machines and to assure that software agents can read web pages easily and carry out sophisticated tasks for users. In fact, "*the Semantic Web is not a separate web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" (Berners-Lee et al., 2001).

The Semantic Web is an effective tool for globalizing knowledge representation and sharing on the web. To create such an infrastructure, two important technologies for developing the semantic web are used: XML (eXtensible Markup Language) and the RDF (Resource Description Framework) where the XML allows users to annotate their web pages by adding arbitrary structure without explaining the meaning which will be expressed by RDF in the form of triples.

The Semantic Web can be viewed as a globally accessible database but with semantics provided for information exchanged over Internet. To introduce the concept of ontology, one of the basic component of the Semantic Web, let's consider the following example. Suppose that we have access to a variety of databases with information about students. If we want to find a student with a particular identifier, his inscription number for example, we need to know in each database the field representing this number to get access to the student's information. A problem may occur when the databases are not using the same identifier. One of them uses the identity number to identify a student. So a program that wants to use information across the two databases has to know that the two terms are being used to mean the same thing, the same student in our case. The solution is to create collections of information called ontologies. By the use of ontologies, the computers are able to manipulate the terms in an effective way that is

meaningful to the user.

With the emergence of the Semantic Web and the use of ontologies, the machines are able now to process the information, to exchange the knowledge with other programs and to answer users queries whose results do not reside on a single web page.

In this section, we gave a general overview of the Semantic Web. The basic components of this infrastructure, namely the ontologies and the two technologies XML and RDF will be described much more in detail in the next subsections.

### 2.2.2 Definitions of Ontologies

Many definitions have been given to the term *Ontology* because there is no universal agreement on the meaning of it. The notion of ontology is a branch of philosophy which considers the nature of being and existence. Later, ontology has been suggested as a reusable model in the artificial intelligence to facilitate information exchange and knowledge sharing.

According to (Gruber, 1993) *"an ontology is an explicit specification of a conceptualization"*. In this definition *conceptualization* characterizes an abstract model in which the concepts related to a particular domain were identified and *explicit specification* refers to an explicit definition of the different concepts, the relationships between them and the constraints on their use related to the abstract model.

Another definition was given by (Borst et al., 1997) who considered an ontology *"as a formal specification of a shared conceptualization"* where *formal* refers to the fact that ontology should be machine readable and *shared* reflects the notion that an ontology contains knowledge used and reused across different applications.

Based on a technical view and following (Uschold and Gruninger, 2004), *" an ontology represents many different kinds of things in a given subject area. These things are represented in the ontology as* ***classes*** *(concepts) and are typically arranged in a lattice or taxonomy of classes and subclasses. Each class is typically associated with various* ***properties*** *(slots, roles) describing its features and attributes as well as various* ***restrictions*** *on them (facets, role restrictions). An ontology together with a set of concrete* ***instances*** *(individuals) of the class constitutes a knowledge base.*

In (Ehrig and Sure, 2004), an ontology O is defined formally as a tuple:

$$\mathrm{O} := (\mathrm{C}, H_C, R_C, H_R, \mathrm{I}, R_I, \mathrm{A})$$

This tuple consists of the following: Concepts $C$ are arranged in a subsumption hierarchy $H_C$. Relations $R_C$ exist between single concepts. Relations can also be arranged in a hierarchy $H_R$. Instances $I$ of a specific concept are interconnected by property instances $R_I$. Additionally, one can define axioms $A$ which can be used to infer knowledge from already existing one.

### 2.2.3 Categories of Ontology Application Scenarios

Ontology is a shared model containing vocabulary related to a specific domain and it is used in order to facilitate the communication between people and applications. Four main categories of ontology application scenarios have been identified by (Uschold and Gruninger, 2004):

- **Neutral Authoring**: An organization can benefit greatly from non-interoperable tools and formats to develop its own neutral ontology for authoring and then develops translators from this ontology to the terminology required by the various target systems. The advantages of this method are knowledge reuse, maintainability and long term knowledge retention.

- **Common Access to Information**: To assure an interoperation and an information sharing between the different agents, ontology can be used as agreed standard to translate between the different formats and representations that evolved independently. This kind of use of ontologies assure interoperability and makes the use of the knowledge more effective.

- **Ontology-Based Specification**: There is a growing interest in the idea of *"Ontology Driven Software Engineering"*. The main idea is to create an ontology that specifies the different things that the software system must address, then this ontology is used as a (partial) set of requirements for building the software. This scenario ensures greater interoperation, facilitates the maintenance and reduces costs.

- **Ontology-Based Research**: In this case, ontology is used as a repository where the information is structured in order to facilitate search. This kind of use supports the organization and classification of repositories of information at a higher level of abstraction than it is commonly used today. This scenario gives a better access to the information.

## 2.3 Ontology Representation

Ontologies play a prominent role in helping applications to get access to the information. In order to assure this, ontologies are supposed to be structured in a formal vocabulary giving the possibility to the applications to explore the content of the information instead of just presenting it to humans.

In order to support the usage of the ontologies in many areas (e-commerce, search engines, web service...), an ontology language **OWL** (Web Ontology Language) was designed as a major formalism for the representation of the information particularly in the semantic web. OWL is considered as a sophisticated language due to the fact that it was not designed in a vacuum. In other words, this language was developed based on several communities including the Description Logics, the RDF and the existing ontology languages which have an influence on the OWL.

### 2.3.1 Influences on OWL

Each of the influences listed above (Description Logics, RDF and ontology languages) will be described in detail in this subsection showing how these communities have an effect on the design of the OWL to make it more expressive than its predecessors of languages.

#### Description Logics

In recent years, Description Logics (DLs) inspire the development of ontology languages and are considered as their logical basics and inference mechanisms. For that purpose, we will define this family of formalisms by stating its origins and its basics, namely its semantic and the reasoning mechanisms.

Description logics are a family of formalisms for knowledge representation and reasoning. In fact, they are used to provide descriptions of the world as well as to enable applications to find implicit consequences of its explicitly represented knowledge.

The DLs are originated from semantic networks and frame-based systems which are two non logic based approaches for representing knowledge. In fact, these network-based systems aim at representing the domain of discourse by using a network which specifies the set of individuals and their relationships. These systems were not satisfactory because of their lack of formal logic based semantics. For that purpose DLs are developed to refer on the one hand, to concept description and on the other hand to the logic based semantics (Nardi and Brachman, 2003).

To describe a domain, DLs use concepts, roles and individuals.

- **Concepts** denote classes of objects. There are two kinds of concepts: *primitive concept* whose instances are determined entirely by the user who provides only necessary conditions and *defined concept* which is specified by giving necessary and sufficient conditions. A primitive concept will be introduced to the system by writing an expression of the form <*identifier*> $\sqsubset$ <*necessary conditions*>. For example the concept "Person" can be a primitive concept and can be introduced as a subclass of another primitive concept "Mammal" by stating that Person $\sqsubset$ Mammal. When there is no specific necessary condition to be expressed, we can use the concept *THING* which is a superconcept of all other concepts in the hierarchy. For example we can state that Mammal $\sqsubset$ THING. On the other side, *NOTHING* is used to express that a concept is a subconcept of all other concepts in the hierarchy. Concerning the defined concepts, they are introduced in the form of: <*identifier*> $\equiv$ <*necessary and sufficient conditions*>.
- **Roles** are used to express binary relationships between two concepts. One of the key characteristic features of DLs resides in the constructs for establishing these relationships. We distinguish between two restrictions: *value restrictions* and the *number restrictions* where the former expresses the limitation of the value of the role's filler and the latter provides minimum and maximum on the number of fillers of a role.

- **Individuals** are generally asserted to be instances of concepts and have roles filled with other individuals.

The semantics of a DL are expressed by an interpretation $I = (\Delta^I, .^I)$ which consists of a domain of values $\Delta^I$ and an interpretation function $.^I$. This function maps every concept to a subset of $\Delta^I$ and maps every role and attribute to a subset $\Delta^I \times \Delta^I$ and every individual to an element of $\Delta^I$. An interpretation I is a model of a concept C if $C^I$ is non empty. An interpretation I is a model of an inclusion axiom $C \sqsubseteq D$ if $C^I \subseteq D^I$.

The main inference task on concept expressions in DLs is *subsumption* typically expressed as $C \sqsubseteq D$. Determining subsumption is the problem of checking whether the concept denoted by D (the subsumer) is considered more general than the one denoted by C (the submee). Another inference task on concept expressions is *concept satisfiability* which is considered as a special case of subsumption and consists of checking whether a concept does not necessarily denote the empty concept. The concept satisfiability can be reduced to subsumption by considering that the subsumer is an empty concept which means that a concept is not satisfiable.

There are two main strategies used in description logics to calculate subsumption and they are detailed in (McGuinness, 1996):

- The first approach, *Structural Translation*, is implemented in two phases: normalization and comparison. In order to determine if A subsumes B, B is first normalized. During normalization, a structured form is completed using information that has been told to the system about B. This told information is used along with rules of inference to try to derive additional information and to eliminate redundant parts. Once B is normalized, the syntactic form of B is compared to the syntactic form of A to check subsumption. The subsumption in this method will easily be sound but hard to be complete due to high computational complexity.

- The second approach, *Tableau Method* aims to make the subsumption complete. In this case, the basic inference consists on determining if a concept is inconsistent or not, and relies on the observation that C is subsumed by D if and only if $C \sqcap \bar{D}$ is not satisfiable, or C is not subsumed by D if and only if there exists a model for $C \sqcap \bar{D}$. This method tries to generate such a finite model by using an approach similar to first-order tableaux calculus with a guaranteed termination. If it succeeds then the subsumption relationship does not hold, if it fails to find a model then the subsumption relationship holds.

Within a knowledge base, we distinguish between *intensional knowledge* (general knowledge about the problem domain) which is usually thought not to change and *extensional knowledge* (specific to a particular problem) which is usually thought to be dependent on a single set of circumstances. A DL knowledge base usually includes two components:

- TBOX (terminological KB) contains intensional knowledge in the form of a terminology. It consists of concepts and roles defined for a domain and a set of axioms used to assert relationships (subsumption, equivalence...). For example, a woman can be defined as a female person by writing this declaration:

$$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$$

- ABOX (assertional KB) contains extensional knowledge , specific knowledge of the individuals of the domain of interest. It includes a set of assertions on individuals by using concepts and roles in TBOX. For example to state that the individual Jeanne is a female person we write:

$$\text{Female} \sqcap \text{Person}(\text{Jeanne})$$

**XML**

HTML was designed as a language to be applied for defining the presentation of the information on the web. But with the increase amount of information available on the web, a need to define the structure of information was desired in order to facilitate automated processing of web content. For that purpose XML has been developed and considered as the basic language for the semantic web. It helps the users to define their own tags to describe the structure of the web documents.

**RDF**

XML is a tag-based language, it describes the structure of the information exchanged on the web but fails to define the semantics in a way understandable by machines. For that reason, RDF has been proposed as a solution to fill up the hole. It is an XML based language defined to add meta-information to the web documents to describe resources (an entire web page, a part of a web page or an object that is not directly accessible via the web). These resources are described through RDF graph composed of RDF triples (subject, predicate, object) as shown in the figure where:

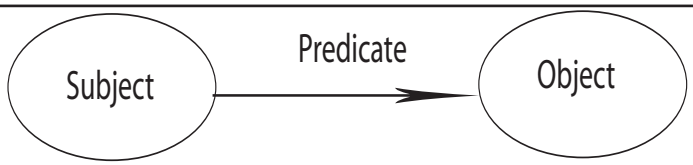


Figure 2.1: RDF graph

- **Subject** is a resource described with or without URI (Uniform Resource Identifier).
- **Predicate** defines attributes or relations used to describe a resource.
- **Object** is a resource described with or without URI or a literal (string or fragment of XML)

### RDFS

It is built on top of the RDF to lift the limit of it. In fact RDF allows modeling objects with defined semantics, in the sense that it models RDF classes and properties hierarchically. It declares subclasses and subproperties relations and provides range and domain constraints. RDFS provides main features for representing knowledge but it is a limited language and a more expressive power is demanded to represent ontological knowledge.

In (Antoniou and Van Harmelen, 2004), the authors listed the main features not expressed in RDFS that should be taken into consideration by an ontology language:

- Local scope of properties: In RDFS we can not declare range restrictions applied to some classes only.
- Disjointness of classes: Sometimes we wish to state that two classes are disjoint. For example, the two classes *bedAndBreakfast* and *hotel* are disjoint but with the RDFS we can only express the subclass relationships, e.g. *bedAndBreakfast* is a subclass of *accommodation*.
- Boolean combination of classes: It allows to build new classes by combining other classes using union, intersection and complement.
- Cardinality restrictions: It gives the possibility to place restriction on how many distinct values a property may or must take. For example, we wish to state that a *destination* has at least one *accommodation* and two *activities*.
- Special characteristics of properties: Giving some more information about properties is interesting to state that a property is transitive, unique or even the inverse of another one.

### OIL and DAML+OIL

OIL [1] (Ontology Inference Layer) and DAML+ OIL [2] are richer languages developed to handle with the limited expressivity of the RDF and RDFS. DAML+OIL was developed as an extension of the RDF(S) with powerful concepts for describing ontologies. Compared to RDF(S), DAML+OIL has some additional features. In fact it allows the classes to be specified as logical combinations of other classes or as enumeration of objects as well as making the description of the properties more specific (transitive, symmetric, functional...), in addition to the possibility of adding cardinality restriction to limit the number of statements with the same subject and predicate. But the major extension over RDFS is that DAML+OIL is able to provide restrictions on properties through the datatypes.

The figure above presents the most important markup languages illustrating the relationships existing between them. As shown, the OWL is an extension of the DAML+OIL

[1] http://www.ontoknowledge.org/oil/
[2] http://www.daml.org/2001/03/daml+oil-index

and is built upon the RDF(S) with more descriptive power.

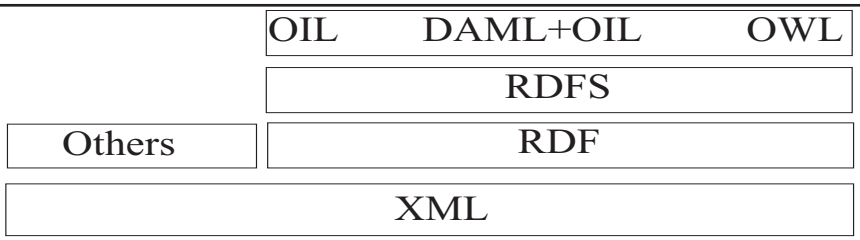


Figure 2.2: XML Markup Languages

This language will be described in detail in the next subsection because it is the most important one for the representation of ontologies and it is also the underlying formal ontology language for our work.

### 2.3.2 Web Ontology Language

The OWL is developed by the World Wide Web Consortium (W3C) Web Ontology Working Group and has to fit into the semantic web vision of existing languages: XML, RDFS, DAML+OIL and to overcome the lack of their expressivity. In fact:

- XML provides a syntax for structured documents without semantics on the meaning of these documents.
- XML Schema is used to restrict the structure of XML documents and extends XML with datatypes.
- RDF is a data model for the objects ("resources") and the relations between them providing simple semantics for this data model.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources.

OWL was designed to add vocabulary about the properties and classes. In an OWL ontology, we can describe classes, properties, individuals and axioms used to associate class and property identifiers with specification of their characteristics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL and OWL Full.

In the following we describe in detail the notion of classes, properties and individuals.

- **Classes**: A class provides a way to put together the different resources with similar characteristics. Every OWL class is associated with a set of individuals (class extension). These individuals in a class extension are called instances of the class.
  OWL classes are described through class description that can be combined into class axioms.

– *Class description*: We distinguish different ways of describing a class. A **named class** is defined by an identifier through *"rdf:ID"*.

```
<owl:Class rdf:ID="Accommodation">
```

An **anonymous class** is represented by placing constraints on the class extension. This second kind describes a class containing exactly the enumerated individuals, or a class of individuals satisfying a property restriction or even a class that satisfies boolean combinations of class descriptions.

* Enumeration: To enumerate the individuals that are instances of a class we use *"owl:OneOf"*

```
<owl:Class>
 <owl:oneOf rdf:parseType="Collection">
  <AccommodationRating rdf:ID="OneStarRating">
  <AccommodationRating rdf:ID="#TwoStarRating">
</owl:oneOf>
</owl:Class>
```

* Property restrictions: It describes an anonymous class, namely a class of all individuals that satisfy the restriction. It is introduced in the OWL ontology through *"owl:restriction"*.

  We depict two forms of restrictions: value constraint and cardinality constraint. The former puts constraints on the range of the property. It can be introduced by *(owl:AllValuesFrom, owl:someValuesFrom, owl:hasValue)*.

```
<owl:Class rdf:ID="City">
  <rdfs:subClassOf>
   <owl:restriction>
    <owl:someValuesFrom rdf:resource = "#LuxuryHotel"/>
  </owl:restriction>
 </rdfs:subClassOf>
</owl:class>
```

  The latter puts constraints on the number of values that a property can take. To allow only a specific number of values for a property, OWL provides three constructors*(owl:maxCardinality, owl:minCardinality, owl:cardinality*).

```
<owl:Class rdf:ID="#Destination">
  <owl:Restriction >
   <owl:onProperty>
     <owl:objectProperty rdf:about="hasAccomodation">
   </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/
    2001/XMLSchemaint">1
   </owl:minCardinality>
  </owl:Restriction>
</owl:Class>
```

* Intersection, Union, Complement: An anonymous class can be described by logical operations on classes by introducing *(owl:intersectionOf, owl:unionOf, owl:complementOf)*

```
<owl:Class rdf:ID="#BudgetHotelDestination">
  <owl:equivalentClass >
   <owl:Class >
   ...
     <owl:intersectionOf rdf:parseType="Collection" >
       <owl:Class rdf:about="#BudgetAccommodation" />
       <owl:Class rdf:about="#Hotel" />
      </owl:intersectionOf >
    </owl:equivalentClass >
   </owl:class >
```

– *Class axioms*: They give more information about the characteristics of a class. OWL contains three language constructors for combining class descriptions into class axioms: *"rdfs:subClassOf"*(the class extension of a class description is a subset of the class extension of another class description)

```
<owl:Class rdf:ID="Sunbathing" >
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Relaxation">
  </rdfs:subClassOf>
</owl:Class>
```

*owl:equivalentClass* (a class description has exactly the same class extension as another class description) and *owl:disjointWith* (a class extension of a class description has no members in common with the class extension of another class description)

```
<owl:Class rdf:ID="Hotel" >
  <owl:disjointWith>
    <owl:Class rdf:about="#Campground">
  </owl:disjointWith>
</owl:Class>
```

- **Properties**: Two main categories of properties can be defined: objectProperties (*owl:objectProperty*) that link individuals to individuals and dataTypeProperties (*owl:dataTypeProperty*) which link individuals to data values. Property axioms can be used to define additional characteristics of properties.

  The following constructors supported by OWL, allow to define property axioms.

  - *RDF Schema constructs*: We note *rdfs:subPropertyOf* an axiom to arrange properties in a hierarchy and mention that a property is a subproperty of another one. In addition to this axiom, *rdfs:domain* is another property axiom that can be defined to link a property to a class description asserting that the subjects of the property must belong to the class extension of the mentioned class description. Finally, *rdfs:range* is defined to link a property to either a class description or a data range.

```
<owl:ObjectProperty rdf:ID="hasContact" >
  <rdfs:range rdf:resource="#Contact">
  <owl:domain rdf:about="#Activity">
</owl:ObjectProperty>
```

  - *Relations to other properties*: We distinguish other constructors used to relate two properties. *owl:equivalentProperty* stating that two properties have the same property extension and *owl:inverseOf* defining inverse relation between properties (in both directions from domain to range or from range to domain)

```
<owl:ObjectProperty rdf:ID="hasActivity" >
  <owl:inverseOf rdf:resource="#isOfferedAt>
  <rdfs:range rdf:resource="#Activity">
  <owl:domain rdf:about="#Destination">
</owl:ObjectProperty>
```

  - *Global cardinality constraints on properties*: There exists property axioms that specify restrictions on property cardinality such as *owl:FunctionalProperty* defines that a property can have only one value y for each instance x and *owl:InverseFunctionalProperty* determines that the object of a property statement uniquely determines the subject (some individual).

```
<owl:FunctionalProperty rdf:ID="hasZipCode">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchemaint"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owlDatatypeProperty"/>
  <rdfs:domain rdf:resource="Contact"/>
</owl:FunctionalProperty>
```

  - *Logical characteristics of properties*: To describe logical features on properties, one can introduce the following property axioms: *owl:TransitiveProperty* and *owl:SymetricProperty*.

- **Individuals**: An individual is introduced by declaring it to be a member of a class. To tie an individual to a class we use *rdf:type*. Some constructors can be used to state the identity of individuals: *owl:sameAs* (used to equate individuals defined in different documents to one another), *owl:differentFrom* (used to state that two references refer to different individuals) and *owl:allDifferent* (used to define that a list of individuals are all different)

```
<AccommodationRating rdf:ID="ThreeStarRating">
 <owl:differentFrom >
  <AccommodationRating rdf:ID="TwoStarRating">
   <owl:differentFrom rdf:resource="#OneStarRating"/>
   <owl:differentFrom rdf:resource="#ThreeStarRating"/>
  <AccommodationRating >
 </owl:differentFrom >
</AccommodationRating >
```

We will be interested in this dissertation on the classes and the constructors related to them, in particular on the named classes introduced in an ontology by *"rdf:ID"* and the anonymous one described by logical operators (*"owl:intersectionOf"*, *"owl:unionOf"*) and the one described through constructors like (*"rdfs:subClassOf"*, *"owl:equivalentClass"*, *"owl:disjointWith"*).

## 2.4 Ontology Reasoning

Ontologies are finding, more and more, their ways in many applications such as e-commerce (McGuinness, 1999), search engines and grid services. Not only they serve as the foundation of the semantic web, but also they are applied in knowledge management systems, information integration...

In fact an ontology is not just used to build a static structure, a taxonomy where different concepts are interrelated to each other but to enhance systems as well with reasoning tasks and to improve queries possibilities in order to infer implicit knowledge. Answering queries over ontology classes and instances helps to find more general classes and to retrieve individuals that match to a given query.

### 2.4.1 Purposes of Reasoning

We mean by reasoning over an ontology any mechanism for making explicit a set of facts that are implicit in an ontology. Many purposes incite to do reasoning tasks, we single out the most important ones:

- **Validation**: As mentioned earlier, an ontology is a representation of the domain that we are modeling. To get a coherent representation and a correct one, an ontology must be valid. Reasoning is very important for validation and it is done for example through a consistency checking which consists on verifying that the ontology is modeling correctly the domain.

- **Analysis**: Once we are sure that the ontology is valid and it is a well representation of the domain of discourse, we can reason over it by deducing facts about the domain which help to collect new information and to infer implicit knowledge.

### 2.4.2 Reasoning Tasks

Many ontology reasoning tasks can be done over an ontology. Among them we mention:

- **Ontology consistency**: Ontology can not be treated as static and the knowledge represented in an ontology should not be considered as fixed because due to changes and modifications that can occur. These modifications concern the domain representation, correction of the design and change in the user's requirement. So these update can change an ontology from a consistent state to an inconsistent one.

  A consistency state should be assured by having reasoners able to detect and to resolve inconsistency. In (Haase and Stojanovic, 2005), the authors distinguish three forms of ontology consistency: structural consistency, logical consistency and user-defined consistency.

  - *Structural consistency*: This notion of consistency ensures that the ontology conforms to the ontology language constraints imposed by this language. Structural consistency can be enforced by verifying a set of structural conditions related to the ontology language in use. As an example of structural conditions we can state *"The complement of a class must be a class"*.
  - *Logical consistency*: An ontology is logically consistent if it does not contain contradicting information, it conforms to the underlying formal semantics of the ontology language.
  - *User-defined consistency*: A lack of definitions of consistency may be not captured by the ontology language which lead to additional conditions defined explicitly by some applications or users to assure the ontology consistency. As an example users could require that classes can only be defined as a subclass of at most one of other class.

- **Concept satisfiability**: It is another typical reasoning task. It verifies whether a concept does not necessarily denotes the empty concept.

- **Concept subsumption**: One of the basic inference task is concept subsumption. The aim of a subsumption is to check whether a concept (the subsumer) is considered more general than another one (the subsumee). In other words, *"subsumption checks whether the first concept always denotes a subset of the set denoted by the second one"*.

- **Concept equivalence**: A concept A is equivalent to concept B if A and B subsume each other.

- **Concept disjointness**: Two concepts are disjoint if they don't have a common instance.

To assure the tasks listed above, a variety of reasoners exists. Among them, we mention: FACT++, RACER, PELLET.

- FACT++ [3]: It is the new generation of the well-known FACT OWL reasoner. It uses the established FACT algorithm but with a different internal architecture. It is implemented using C++.

- RACER [4]: It is an optimized tableau reasoner for the DL SHIQ(D). Besides basic reasoning tasks as satisfiability and subsumption, it offers the possibility to answer to queries. It is implemented in the Common LisP Programming Language.

- PELLET[5]: It is a Java based OWL DL reasoner. It provides functionality to validate ontology species, check consistency of ontologies, classify the taxonomy, check entailments and answer queries.

## 2.5 Ontology Mapping

Ontologies are considered as a key factor to enable interoperability among heterogeneous systems and semantic web applications. In fact tasks are located on distributed and heterogenous systems that demand support from more than one ontology. To assure the communication between these information sources, a mapping between their ontologies must be built.

Ontology mapping, also called ontology alignment is defined as *"the task of establishing a collection of binary relations between the vocabulary of two ontologies."* (Kalfoglou and Schorlemmer, 2003)

According to Ehrig and Staab, the mapping can be defined more formally by the fact of *"Given two ontologies $O_1$ and $O_2$, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology $O_1$, we try to find a correspondence entity which has the same intended meaning in ontology $O_2$"*(Ehrig and Staab, 2004).

### 2.5.1 Categories of Ontology Mapping

Depending on the application to be held, the authors in (Choi et al., 2006), depicted three main categories for ontology mapping:

- **Mapping between an integrated global ontology and local ontologies:** In this case the global ontology contains a shared vocabulary for the specification of the semantics and all the local ontologies are related to the global one. This category maps a concept found in one ontology over other ontologies.

[3] http://owl.man.ac.uk/factplusplus
[4] http://www.racer-systems.com
[5] http://www.mindswap.org/2003/pellet/index.shtml

- **Mapping between local ontologies:** In this category each information source is represented by its own ontology. Each source can be developed without respect to other sources. In this kind of mapping changes over sources are simple and can be done easily.
- **Mapping in ontology merge and alignment:** In this case, the ontology mapping establishes correspondence among local ontologies to be merged or aligned, and determines the sets of overlapping concepts, synonyms, or unique concepts to those sources.

### 2.5.2 Ontology Mapping Process

In (Ehrig and Staab, 2004), the ontology mapping process is introduced. It subsumes all the mapping approaches as presented on Figure 2.3. This process has as input two ontologies to be mapped. In the following we describe each step mentioned in the process.

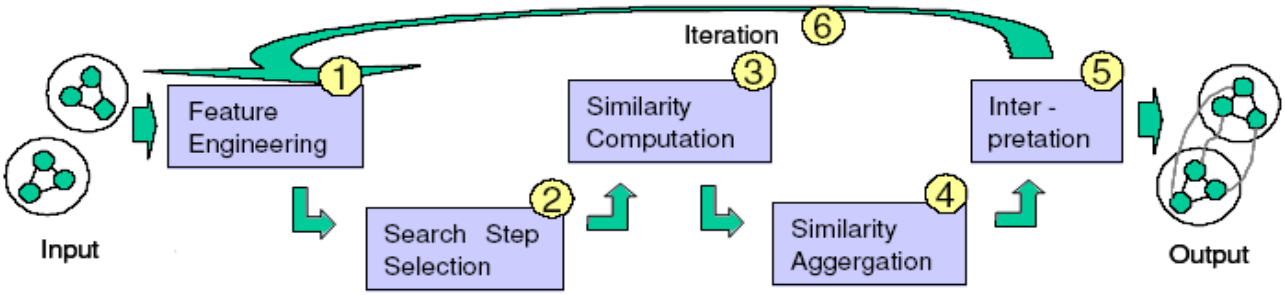


Figure 2.3: A Canonical Mapping Process Inspired by CRISP-DM.

1. **Feature engineering:** The initial representation of ontologies are transformed into a format more adequate for the similarity calculations.
2. **Selection of next search steps:** A restricted subset of candidate concepts pairs is chosen in order to compute the similarity.
3. **Similarity computation:** Determination of the similarity values between candidate mappings based on their definitions in ontologies.
4. **Similarity aggregation:** The aggregation of the different similarity values for one candidate pair into a singled aggregated similarity value.
5. **Interpretation:** Derives mapping between entities using for that the individual or aggregated similarity values.
6. **Iteration:** Several algorithms perform an iteration over the whole process in order to bootstrap knowledge.

In this master, we will focus on the following points:

- Concerning the ontology engineering tasks, we concentrate on the ontology representation and ontology reasoning. We will consider the ontologies written in OWL language because it is the most sophisticated one.
- A special focus will be devoted to some of the OWL constructors especially those related to the classes of an ontology.
- We will be interested in outlining the similarities existing between the OWL ontology and the graphical models used for representing domains of discourse.

## 2.6 Conclusion

In this chapter we presented in detail the ontology research area which is knowing evolution day after day. We focused on three main concerns: the ontology representation by defining the main languages used to represent the ontology and in particular the OWL, the ontology reasoning by defining the main reasoning tasks that can be held across the ontology and finally we gave a brief introduction to the ontology mapping. OWL is a sophisticated language but is unable to represent partial information, so dealing with uncertainty for representing ontologies is an issue to be addressed. Taking into account the uncertainty aspect and its importance in the semantic web field will be discussed in detail in the next chapter.

# 3

# Uncertainty for the Ontology Research

## 3.1 Introduction

The new vision of the web aims at enabling applications to share information which is structured in a machine understandable way. To fulfill this task, ontologies have proved to be a powerful tool and a successful one to capture the knowledge about concepts and their relations. As mentioned in chapter 2, many ontology definition languages have been developed to define ontologies in a formal way. Among them we mentioned the OWL based on crisp logic where a sentence in OWL is either true or false and nothing in between. However, real domains contain uncertainty knowledge or imprecise information. Taking into account these uncertainties in the ontology is crucial.

This chapter is devoted to the uncertainty for the web ontology language. It concentrates on describing some works proposed for extending OWL web ontology language with uncertainty theories (probability theory and fuzzy sets theory). Some works do not focus only on this extension but try also to give an approach for the construction of a graphical model showing how the uncertain information is represented in order to do reasoning tasks later.

This chapter is organized as follows: We first introduce the different models used to express uncertainty (section 3.2), a special concern will be devoted to the probabilistic graphical models (section 3.3). A description of some works extending OWL web ontology language is given in section 3.4 and section 3.5 where the former concentrates on those dealing with the probability theory and the latter with the fuzzy sets theory.

## 3.2 Models for Handling Imperfect Knowledge

In (Smets, 1997), the author considers *imperfection* as the most general label that includes *imprecision*, *inconsistency* and *uncertainty* which are the major forms of imperfect data.
Focusing on the main differences between these terms, Smets precises that imprecision and inconsistency are related to the content of information whereas the uncertainty is

induced by lack of information.

In fact, imprecision *"covers cases where the value of a variable is given but not with the precision required"* (Smets, 1991). We note two main categories of imprecision: imprecision without error and imprecision combined with error. In the former, we evoke vagueness (referring to something not clearly explained or expressed that can be understood in different way) and missing data (something which is lost, not present). In the latter, the data can be incorrect, inaccurate or invalid. The second aspect of imperfection, inconsistency, describes information with parts that are contradicting each other.

Finally, uncertainty *"is partial knowledge of the true value of the data. It results in ignorance (etymologically not knowing). It is essentially, if not always, an epistemic property induced by a lack of information. A major cause of uncertainty is imprecision in the data"*(Smets, 1997). One can distinguish between objective and subjective uncertainty. The objective uncertainty is linked to the information itself whereas the subjective one depends on the observer's opinion about the true value of the data.

The imperfection with its various aspects (imprecision, inconsistency and uncertainty) must be taken into account to guarantee a complete representation and an accurate one of the real world. For this purpose, a model must be used to represent imperfect knowledge. An appropriate one has to be chosen carefully not in a random way because every context has its own model representing it.

In the last years, many new models have been developed to deal with imperfection. Although the well-known one, the probability theory, is successful in many applications, it is not the unique one that can cope with all the aspects of imperfection. Among these models to handle the imperfect knowledge - in addition to the probability theory -we mention the theory of belief functions (Shafer, 1976) proposed to handle uncertainty, the theory of fuzzy sets (Zadeh, 1978) and the theory of possibility (Dubois and Prade, 1988) which are devoted to model imprecision.

In this master, we will use the theory of belief functions as a model to cope with uncertainty in ontology modeling and reasoning. The basic concepts of this theory will be presented in the next chapter.

## 3.3 Probabilistic Graphical Models

As mentioned before, we are interested in modeling ontologies taking into account the uncertain information. To assure this, a formalism should be used to capture the information, make it more explicit and represent it in order to enable ontology reasoning tasks later.

As a formalism, many researchers have suggested graphical models. A graph model is defined as a mathematical structure that specifies the different connections between

the variables. It is considered also as a picture that provides an intuitive description of the problem. Graphical models are powerful in representing problems by transforming the complex one into an easily, clear and well understood representation.

A well-known graphical model, the Bayesian network has been used so much for representing uncertain knowledge. A detailed description of this network will be given in the next subsections.

### 3.3.1 Bayesian Network Definition

The Bayesian networks, also called Bayesian belief networks, belief networks or probabilistic causal networks, are widely used for representing knowledge under uncertainty (Pearl, 1988). Before defining the Bayesian networks formally, some basic concepts from graph theory are reviewed.

A graph **G** is a pair (**V**,**E**) where $\mathbf{V} = \{X_1,...,X_n\}$ is a non-empty finite set of variables, called *vertices* or *nodes* and **E** is a set of some pairs of nodes in **V** called *edges*.

We distinguish between *undirected* graph and *directed* graph. The former is a graph where an arc is drawn between two nodes without a causality meaning between the two of them and the latter is a graph where an arc from one node to another can be informally interpreted as indicating that the first node causes the second. This kind of graphs are more appropriate for representing conditional relations.

Among the directed graphs, we mention the *DAG* (directed acyclic graph) which is a directed graph without any cycles (a cycle is defined as a path starting and ending with the same node).

These elements facilitate to define the Bayesian networks. A Bayesian network is a graph where:

- A set of random variables makes up the nodes of the network.
- A set of directed links connects pairs of nodes.
- Each node has a Conditional Probability Table (CPT) that quantifies the effects that the parents have on the node.
- The graph is a DAG.

In addition to this definition provided by (Russel and Norvig, 1995), the Bayesian network can be viewed as a graph representing two major information: a qualitative information and a quantitative one (Williamson, 2005). Williamson considers the Bayesian network as a graph which consists of a qualitative component and a quantitative one.

- The qualitative component is a DAG G = (V,E) where V and E are respectively the vertices and directed edges in the graph.

- The quantitative component contains a set of probability specifications P. For each variable X ∈ V, P specifies the probability distribution of X.

Each of these two components will be detailed in the following.

### 3.3.2 Qualitative Level

The qualitative level describes how the information is structured on a Bayesian network. In fact, the information is represented by a DAG where the nodes correspond to variables and the edges reflect the existence of direct causal influences between the linked variables.

#### Evidences

Evidence is information about a certain situation via observation. A node is an evidence node if the variable represented by this node is observed. There are three types of evidences that can be applied to a Bayesian network:

- **Hard Evidence**: It is an instantiation of a variable $X_i$ (with $X_i = x_i$). It means that a node $X_i$ is instantiated to a particular state $x_i$ with a probability value equal to 1 and with a probability value of 0 in all other states (Butz and Fang, 2005).
- **Soft Evidence**: It means that E ≠ e. In other words, the set E of variables does not take on value e (Butz and Fang, 2005).
- **Virtual Evidence**: It is the likelihood of a variable's distribution. The likelihood is presented by the probability of observing $X_i$ being in state $x_i$ if its true state is $x_i'$ (Ding, 2005).

#### d-separation

Interdependencies between variables in a Bayesian network can be determined by the network topology. It is illustrated by the notion of *d-separation* which shows how a change of certainty in one variable may change the certainty for other variables. We depict three types of connections in the Bayesian networks: serial connection, diverging connection and converging one as shown in the following figure.

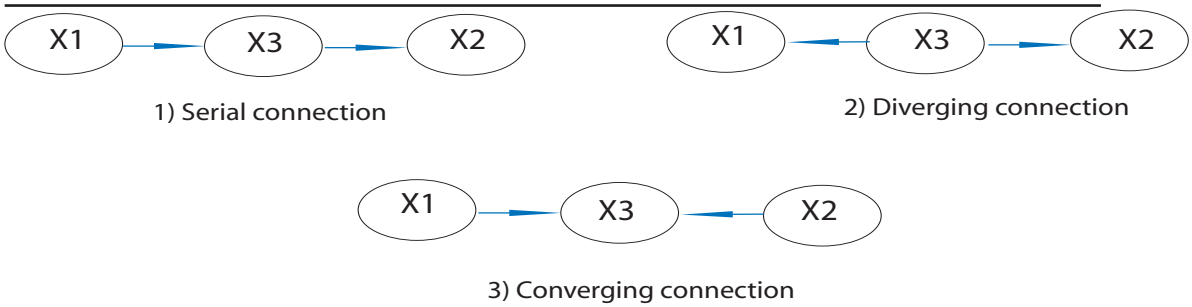


Figure 3.1: Serial, diverging and converging connections in a DAG

- **Serial connection**: The influence of an evidence e can be transmitted unless any variable in the connection is instantiated. ($X_1$ and $X_3$ are d-separated by $X_2$)
- **Diverging connection**: The influence of an evidence e can only be transmitted to the parents of $X_1$ unless its state is known.
- **Converging connection**: The influence of an evidence e can only be transmitted to the parents of $X_1$ if either $X_1$ or one of its descendants has received an evidence (is instantiated).

If two nodes $X_1$ and $X_2$ are *d-separated* by a set of variables, then a change of certainty of $X_1$ have no impact on the belief of $X_2$. In other words, $X_1$ and $X_2$ are independent of each other given the set of variables. So the belief on a node $X_i$ is only influenced by its parents, its children, and all the variables sharing a child with $X_i$.

### 3.3.3 Quantitative Level

We focused on the previous subsection on the network structure, how the information is structured (the variables and the relations between them). The quantitative level describes how the variables in a Bayesian network are related to each other quantitatively through the probability distributions (prior and conditional).

#### Prior Probability

The prior probability can be viewed as a subjective assessment of an expert. It describes what is known about a variable X in the absence of any other evidence.

The prior probability of a variable X independent on any other variables is expressed as P(X). The prior probability of a set of variables $X_1$,...,$X_n$ in a conjunction is represented as P($X_1$,...,$X_n$) and expressed by the *joint probability distribution* (JPD) of these variables.

**Conditional Probability**

Conditional probabilities are used when some evidence has an influence on the variables of a Bayesian network. If we have two variables X and Y and X depends on Y then the conditional probability is written as P(X | Y) which can be expressed by the application of the Bayes' theorem as:

$$\mathrm{P(X \mid Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

In order to represent the quantitative information of P(X | Y), the *Conditional Probability Table* (CPT) is introduced and describes the uncertainty of the causal relationships. A conditional independence assumption is made for the Bayesian networks:

$$\mathrm{P}(X_i \mid \pi_i, \mathrm{S}) = \mathrm{P}(X_i \mid \pi_i)$$

where S is any set of variables excluding $X_i$, $\pi_i$ and all descendants of $X_i$.
Under this assumption, the graphical structure of the Bayesian network allows a representation of interdependencies between variables. The JPD introduced in the prior probability subsection can be obtained through multiplying the conditionals of each variable as follows:

$$\mathrm{P}(X_1,...,X_n) = \mathrm{P}(X_1,...,\mathrm{X}_{n-1})...P(\mathrm{X}_2 \mid X_1)\mathrm{P}(X_1)$$

This distribution can be formulated by applying the following *chain rule*:

$$\mathrm{P}(X_1, \ldots, X_n) = \prod_i P(\mathrm{X}_i \mid \mathrm{Pa}(X_i))$$

where $\mathrm{Pa}(X_i)$ is the parent set of $X_i$.

### 3.3.4 Bayesian Reasoning

The main purpose of constructing a Bayesian network is to use it for reasoning, by computing the posterior probability distribution for query variables by given observations. There are two main types of BN inference tasks: *belief updating* (also called probabilistic inference) and *belief revision* (MAP explanation)(Guo and Hsu, 2002). These tasks start with evidence e but differ on what are to be inferred.

- **Belief updating**: Its objective is to calculate the posterior probability P(X | E) for query nodes X given some observed values of evidence node E.
- **Belief revision**: Its aim is to find the most probable instantiation of some hypothesis variable given the observed evidence. Belief revision for the case when the hypothesis variables are all non-evidence nodes is known as *most probable explanation (MPE)*. An explanation for the evidence is defined as a complete assignment $\{X_1 = x_1,...,X_n = x_n\}$ that is consistent with evidence E. Computing a MPE means finding an explanation such that no other explanation has higher probability.

Researchers have developed exact algorithms and approximate ones to these reasoning tasks. A brief overview is given below and for a detailed survey readers may refer to (Guo and Hsu, 2002).

- **Exact Reasoning**: The approaches of exact reasoning are based on exploration of the causal structures in Bayesian networks to assure an efficient computation. There are various categories of exact inference algorithms. Among them we mention: the polytree algorithm (Pearl, 1988) which is based on local message passing and works for polytrees (single connected BNs). This algorithm is exact and its complexity is polynomial. The polytree algorithm has been extended to work for multiply connected with additional processing: the clustering. It transforms a multiply connected network into a clique by clustering the triangulated moral graph of the underlying undirected graph, then performs message propagation over the clique tree (Lauritzen and Spiegelhalter, 1988). The algorithm's complexity is exponentional in the size of the largest clique of the transformed undirected graph.

- **Approximate Reasoning**: Faced with the problem of applying the exact algorithms with large complex networks, approximate inference algorithms have been proposed. They aim at giving fast, accurate approximations to posterior probabilities in Bayesian networks by reducing the time and space complexity. We list some of the approximate reasoning algorithms: stochastic simulation algorithm and loopy belief propagation.

### 3.3.5 Features of Bayesian Networks

The Bayesian networks possess the following features (Bruyninckx, 2002):

- **Consistency**: The reasoning results in Bayesian networks do not change and do not depend on how the available information is processed.

- **Unique**: Bayesian networks lead to unique conclusions. From the prior knowledge, only one result can be obtained.

Another features have been selected. They underly the major capabilities of the Bayesian networks, what they can offer to the user. In (Glymour, 2001), three utilities are mentioned:

- **Predication**: The use of bayesian networks for predication is clear due to the fact that they can do forward reasoning.

- **Control**: It is referred also as diagnosis. The underlying technique is backward reasoning.

- **Discovery**: The structure of the network for representing a domain application can be discovered from experiments, observations, data and backward knowledge.

## 3.4 Probabilistic Web Ontology Languages

To handle uncertainty in semantic web, two different directions of researches exist. The first carries about representing probabilistic information using an OWL or RDF(S) ontology by proposing a special vocabulary for representing this kind of information. In this category we will be interested in presenting Fukushige's approach (Fukushige,

2005). The second direction of works concerns those interested in uncertainty modeling and reasoning by extending DLs with Bayesian networks. It includes frameworks that, in addition to representing probabilistic information, they use a graphical model to do reasoning tasks. We will present the following frameworks:

- BayesOWL (Ding, 2005): It augments and supplements OWL for representing and reasoning with uncertainty based on Bayesian networks.
- OntoBayes (Yang and Calmet, 2005) : It uses OWL as the underlying ontology modeling language and extends it with additional annotations according to the semantics of the Bayesian networks.
- PR-OWL (Costa et al., 2005): It extends OWL by using multi-entity Bayesian network.
- OWL-QM (Pool et al., 2005): It extends OWL to support the representation of probabilistic relational models (PRM).

These different researches are the main focus of this section.

### 3.4.1 BayesOWL

#### Definition

BayesOWL (Ding, 2005) is a probabilistic framework proposed to deal with uncertainty in the ontology engineering tasks (ontology modeling, ontology reasoning, ontology mapping). This framework has been used first in a single ontology showing how to represent uncertainty knowledge by translating an ontology into a Bayesian network and how to support ontology reasoning tasks across ontologies as probabilistic inferences. To guarantee an interoperability between applications and an information sharing, uncertainty is becoming more prevalent in concept mapping between two ontologies. For this purpose a methodology based on BayesOWL was suggested for automatic ontology mapping.

#### Objectives

The proposed framework tried to overcome some issues that can occur during the ontology engineering tasks (Ding et al., 2005).

- In *ontology modeling*, one can be interested not only in knowing if a class A is a subclass of B but also to have information about the degree of overlap or inclusion between the two classes.
- In *ontology reasoning*, the main problems to be solved are how close two concepts are to each other and the degree of similarity between them even if they are not subsumed by each other. In addition to that, another issue may occur on how to improve the over generalization caused by inputting noisy and uncertain description to an ontology reasoner.

- Dealing with uncertainty in *concept mapping* between ontologies is crucial, in fact a concept defined in one ontology can only find partial matches to one or more concepts in another ontology.

To overcome these issues and to deal with uncertainty, BayesOWL was suggested as a framework that converts an OWL ontology into a Bayesian network.

The choice of this probabilistic model is justified by two main reasons:

- It is an efficient model to enable probabilistic reasoning capability because the Bayesian networks support any inference in the joint space due to the joint probability distribution.

- The DAG of a Bayesian network and the RDF graph of an OWL ontology are similar. In fact both of them are directed graphs and direct correspondence exists between nodes and arcs (Ding et al., 2005).

**The framework**

First, BayesOWL is proposed to address representation and reasoning with uncertainty within a single ontology.

**A- Representation**
The ontology is extended by additional classes to markup probabilistic information. Then, a directed acyclic graph is constructed from the modified ontology. Finally, once the network structure is obtained showing the different nodes and the relations between them, conditional probability tables are built for each node.

In the following we describe each of these steps in detail.

1. **Encoding probability in ontology**

   Web ontology language OWL is based on crisp logic and it can not handle the uncertainty aspect. To add this uncertain information, ontology has to be represented as probability distributions (prior probability, conditional probability), referred also as probabilistic constraints.

   In (Ding and Peng, 2004), the authors provided a solution to make these probabilistic constraints expressed in the form of owl statements. In fact classes in an ontology will be considered as random variables of two states (True and False). For example, a class A will be treated as a random binary variable of two states a and $\bar{a}$. P(A=a) is interpreted as the prior probability that an arbitrary individual belongs to class A and P(a|b) as the conditional probability that an individual of class B also belongs to class A.

   These two kinds of probabilities are encoded by adding two owl classes *"PriorProb"* and *"CondProb"*. A prior probability is defined as an instance of class

*"PriorProb"* which has two properties *"hasVariable"* and *"hasProbValue"*. A conditional probability is defined as an instance of class *"CondProb"* which has three properties *"has Condition"*, *"hasVariable"* and *"hasProbValue"*.

2. **Structural translation rules**

We distinguish two main aspects in a Bayesian network: the qualitative aspect and the quantitative one. In this subsection, we will focus on the qualitative level, i.e. how the network is constructed. In fact the ontology is converted into a network structure (a directed acyclic graph) by following set of structural translation rules.

The general principle underlying these rules is that all classes specified as *"subject"* and *"object"'* in RDF triple are translated into nodes in the Bayesian network and if the two classes are related by *"a predicate"* then an arc is drawn between the two nodes.

Two kinds of nodes are represented in a DAG: *Concept nodes* for concept classes and *L-nodes* which are created during the translation modeling relations between the class nodes that are related by OWL logical operator.

The set of structural translation rules proposed by (Ding et al., 2004) focuses on *class axioms* (defined by rdfs:subClassOf, owl:equivalentClass and owl:disjointWith) and *logical relations* among the concept classes (defined by owl:unionOf, owl:intersectionOf, and owl:complementOf).

These rules are summarized as follows:

- Every concept class is mapped into a binary variable in the Bayesian network.
- Constructor *"rdfs:subClassOf"* is represented by a directed arc from the parent super class node to the child subclass node.
- A concept class C defined as the intersection of concept classes $C_i$(i= 1,$\cdots$,n) using constructor *"owl:intersectionOf"* is mapped into a subnet in the translated Bayesian network with one converging connection from each $C_i$ to C and one converging connection from C and each $C_i$ to an L-Node called **LNodeIntersection**.
- A concept class C defined as the union of concept classes $C_i$(i= 1,$\cdots$,n) using constructor *"owl:unionOf"* is mapped into a subnet in the translated Bayesian network with one converging connection from C to each $C_i$ and one converging connection from C and each $C_i$ to an L-Node called **LNodeUnion**.
- If two concept classes $C_1$ and $C_2$ are related by constructors *"owl:complementOf"*, *"owl:equivalentClass"* or *"owl:disjointWith"*, then an L-Node (na-

med **LNodeComplement**, **LNodeEquivalent**, **LNodeDisjoint**, respectively) is added to the translated Bayesian network, and there are directed links from $C_1$ and $C_2$ to this node.

3. **CPT construction**

In this subsection, we will focus on the quantitative aspect on how to assign conditional probability tables (CPT(s)) to each node once the DAG is constructed.

- **CPT for L-Nodes**
  The CPT for an L-node is determined by the logical relation it represents so that when its state is *"true"*, the corresponding logical relation holds among its parents. We recall that there are five types of L-Nodes: "LNodeComplement", "LNodeDisjoint", "LNodeEquivalent", "LNodeIntersection", "LNodeUnion". The corresponding CPTs related to these logical operators are built as follows.
  - **LNodeComplement**: The complement relation between two concepts $C_1$ and $C_2$ can be obtained by "LNodeComplement = True iff $c_1\bar{c_2} \vee \bar{c_1}c_2$"

Table 3.1: CPT of LNodeComplement

| C1 | C2 | True | False |
|---|---|---|---|
| True | True | 0.000 | 1.000 |
| True | False | 1.000 | 0.000 |
| False | True | 1.000 | 0.000 |
| False | False | 0.000 | 1.000 |

  - **LNodeDisjoint**: The disjoint relation between two concepts $C_1$ and $C_2$ can be realized by "LNodeDisjoint = True iff $c_1\bar{c_2} \vee \bar{c_1}c_2 \vee \bar{c_1}\bar{c_2}$"

Table 3.2: CPT of LNodeDisjoint

| C1 | C2 | True | False |
|---|---|---|---|
| True | True | 0.000 | 1.000 |
| True | False | 1.000 | 0.000 |
| False | True | 1.000 | 0.000 |
| False | False | 1.000 | 0.000 |

  - **LNodeEquivalent**: The equivalence relation between two concepts $C_1$ and $C_2$ can be realized by "LNodeEquivalent = True iff $c_1c_2 \vee \bar{c_1}\bar{c_2}$"

Table 3.3: CPT of LNodeEquivalent

| C1 | C2 | True | False |
|---|---|---|---|
| True | True | 1.000 | 0.000 |
| True | False | 0.000 | 1.000 |
| False | True | 0.000 | 1.000 |
| False | False | 1.000 | 0.000 |

– **LNodeIntersection**: The relation that C is the intersection between two concepts $C_1$ and $C_2$ can be obtained by "LNodeIntersection = True iff $cc_1c_2 \vee \bar{c}\bar{c_1}c_2 \vee \bar{c}c_1\bar{c_2} \vee \bar{c}c_1\bar{c_2}$"

Table 3.4: CPT of LNodeIntersection

| C1 | C2 | C | True | False |
|---|---|---|---|---|
| True | True | True | 1.000 | 0.000 |
| True | True | False | 0.000 | 1.000 |
| True | False | True | 0.000 | 1.000 |
| True | False | False | 1.000 | 0.000 |
| False | True | True | 0.000 | 1.000 |
| False | True | False | 1.000 | 0.000 |
| False | False | True | 0.000 | 1.000 |
| False | False | False | 1.000 | 0.000 |

– **LNodeUnion**: The relation that C is the union of two concepts $C_1$ and $C_2$ can be realized by "LNodeUnion = True iff $cc_1c_2 \vee c\bar{c_1}c_2 \vee cc_1\bar{c_2} \vee \bar{c}c_1\bar{c_2}$"

Table 3.5: CPT of LNodeUnion

| C1 | C2 | C | True | False |
|---|---|---|---|---|
| True | True | True | 1.000 | 0.000 |
| True | True | False | 0.000 | 1.000 |
| True | False | True | 1.000 | 0.000 |
| True | False | False | 0.000 | 1.000 |
| False | True | True | 1.000 | 0.000 |
| False | True | False | 0.000 | 1.000 |
| False | False | True | 0.000 | 1.000 |
| False | False | False | 1.000 | 0.000 |

The situation in which all the L-Nodes in the translated Bayesian network are true, states as $\tau$, so the CPTs for concept nodes should be constructed in the subspace of $\tau$ to get a joint probability distribution consistent with all the given probabilistic constraints.

- **CPT for concept nodes**

  A method based on IPFP (Iterative Proportional Fitting Procedure (Vomlel, 1999)), SD-IPFP algorithm (Ding, 2005) is used to approximate the CPTs for the concept nodes. This algorithm tries to overcome some difficulties that may occur when trying to determine the CPTs.
  The CPTs for the concept nodes are in the form of P($X_C$ $|\tau$) which should be consistent with the probabilistic constraints.

  Two main difficulties may occur:

  – The constraints are not given in the form of CPT.
  – CPTs are given in the general space of X = $X_C \cup X_L$ but constraints are for the subspace of $\tau$ (Notice that $X_C$ is a set for the concept nodes and $X_L$ a set for the L-Nodes) .

  The goal is to construct CPTs for each node $C_i$ in $X_C$ such that the joint probability distribution of $X_C$ in the subspace of $\tau$ is consistent with all the given constraints and Q($X_C|\tau$) to be as close as possible to the initial distribution.

  To construct the CPT of the L-Nodes, the algorithm SD-IPFP should be applied as follows:

  $$Q_{(k)}(C_i|\pi_{Ci}) = Q_{(k-1)}(C_i|\pi_{Ci}).\frac{R(C_i|O_{C_i})}{Q_{(k-1)}(C_i|O_{Ci},\tau)}.\alpha_{(k-1)}(\pi_{Ci})$$

  where $\alpha_{(k-1)}(\pi_{Ci}) = Q_{(k)}(\tau)/Q_{(k-1)}(\tau)$ is the normalization factor.

**B- Reasoning**

In addition to the ontology modeling, the BayesOWL supports the ontology reasoning tasks as probabilistic inferences in the translated Bayesian network.

The most important ones are listed in the following:

- **Concept satisfiability** checks if a concept represented by a description e exists by determining if P(e| $\tau$) = 0.
- **Concept overlapping** determines the degree of the overlap or inclusion of a description e by a concept C which is measured by P(e|c,$\tau$).
- **Concept subsumption** finds the most similar concept C to a given description e defining for that a similarity measure between e and C based on Jaccard coefficient (Van Rijsbergen, 1979).

**B- Mapping concepts**

It is becoming essential to map between concepts in ontologies to assure the semantic integration between the different applications. In (Pan et al., 2005), the authors developed a methodology to assure the ontology mapping. The idea is based on a way to allow two Bayesian networks to exchange beliefs via variables that are similar but not identical. The BayesOWL aims at providing a better solution to other approaches proposed by:

- Describing how the mapping shall be done for pair similar concepts.
- Allowing the mapping between two variables (A in a first Bayesian network and B in the second one via semantic linkage).
- Permitting two kinds of mapping "1 to n" mapping and "m to n" mapping where the former maps one variable in one Bayesian network to multiple similar ones in another one and the latter is used when multiple variables in one Bayesian network need to be mapped.

### 3.4.2 OntoBayes

**Definition**

Due to the fact that agents act in a dynamic and open environment which is characterized by an incomplete and imprecise information, OntoBayes (Yang and Calmet, 2005) has been proposed to enable them to act under uncertainty.

OntoBayes is an ontology-driven uncertainty model which represents uncertain information in a Bayesian network structure by extending OWL and annotating it with Bayesian probability as well as dependency relations.

In (Yang and Calmet, 2005), the researchers justified their choice of the Bayesian network by the fact that it is the most well known graphical model for the representation of probabilistic knowledge but it can not be used to represent knowledge in complex domains for this purpose OntoBayes has been suggested. And the ontological model is unable to express the domain specific uncertainty.

**Knowledge representation**

The OntoBayes focuses especially on the knowledge representation under uncertainty and it follows three main steps. The model is constructed by first integrating probabilities in OWL, then specifying the dependency relations to finally construct the model. We summarize in the following the different steps:

1. **Probability annotated OWL**

   To make the ontology able to express uncertainty information, the OWL has to be extended by specifying probability annotated classes.

To reach this goal, three classes were defined:

- "PriorProb" identifies the prior probability and has a datatype property *"probValue"*
- "CondProb" identifies the conditional probability and has a datatype property *"probValue"* which expresses the probabilistic value.
- "FullProbDist" identifies the full disjoint probability distribution and has two object properties*"hasPrior"* and *"hasCond"*.
  The former specifies the relation between classes *"FullProbDist"* and *"PriorProb"* and the latter specifies the relation between classes *"FullProbDist"* and *"condProb"*.

2. **Dependency annotated OWL**

   Once the random variables are specified and annotated with probabilities, dependency relations between these variables have to be identified.
   To markup the dependency information, an additional property element<*rdfs:dependsOn*> is added to the ontology.
   The random variables specified in an OntoBayes are either datatype property or object property and the dependency is specified between these properties and not between classes to avoid errors and confusions that may occur.

3. **Graphical representation of OntoBayes**

   Once the probabilistic information is added to the OWL and dependency relations between properties are determined, the OntoBayes can be represented graphically. Two kinds of graphs are represented: the *owl graph* and the *Bayesian graph*. The former is a directed graph visualizing classes and relations between them and built on the RDF graph. The latter shows clearly the dependency relations between properties and is extracted from the owl graph by representing separately all the dependency triples from an OntoBayes ontology and then all the triples will be merged.

### 3.4.3 Other Related Works

#### PR-OWL

In order to handle with the lack of a principled means to represent and reason about uncertainty, a probabilistic ontology approach PR-OWL has been proposed.

PR-OWL (Costa et al., 2005), the probabilistic ontology language, is implemented based on the probabilistic ontologies and uses the MEBN (*Multi-Entity Bayesian networks*), as its underlying logic basis. In fact, the probabilistic ontologies are used to describe knowledge about a domain and the uncertainty regarding that knowledge in a principled, structured and sharable way that can be read and processed by computer.

A MEBN is a formalism for representing knowledge by combining first order logic with Bayesian probability. The figure above gives an overview of a PR-OWL MEBN Theory concepts.

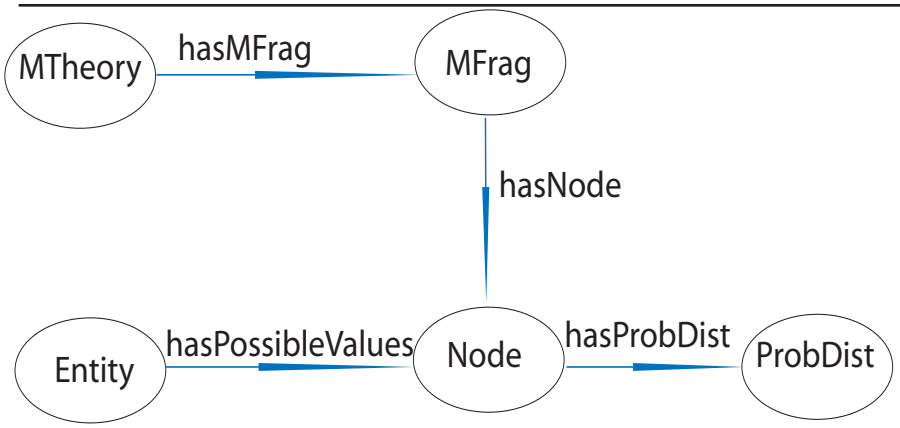


Figure 3.2: Overview of a PR-OWL MEBN Theory Concepts.

MEBN logic represents the world as a set of entities with attributes related to other entities. The features of entities and the different relationships between them are represented through random variables. A probabilistic ontology consists of at least one individual of class MTheory that is formed by a collection of MFrags. Individuals of class MFrag represent the nodes which can be input, resident and context. Like in Bayesian networks, each individual of class Node is a random variable and has a set of possible states. The object property hasPossibleValues links each node to its possible states, which are individuals of class Entity. The random variables have unconditional or conditional probability distributions which are represented by class ProbDist and linked to its respective nodes via the object property hasProbDist. In order to specify the conditional probabilities, the PR-OWL uses the class CondRelationship which expresses the n-ary relation because in OWL only binary relations can be directly constructed.

According to (Costa et al., 2006), the PR-OWL provides the following key features:

- The PR-OWL is compatible with OWL DL. It makes use of additional language markups at the level of OWL DL to integrate the probability into ontologies.
- The expressivity of MEBN helps to model complex problems with Bayesian networks.
- The PR-OWL is flexible. In fact it can be used for different Bayesian probabilistic tools based on different probabilistic technologies.

### OWL-QM

OWL-QM (*OWL Quiddity*Modeler*) (Pool et al., 2005) is an extension of OWL for eliciting and representing PRM (*Probabilistic Relational Model*) which is a model that aims to express the uncertainty about the values of attributes of objects.

In fact a PRM is based on a relational schema which is a set classes C = $\{C_1,...,C_n\}$, where each class $C$ is associated with a set of propositional attributes *A(C)* and a set of relational attributes or reference slots *R(C)*. An instantiation $I$ of a schema is a set of objects, each object belonging to some class C with all propositional and relational attributes of each object specified. A PRM encodes a probability distribution over the set of all possible instantiations I of a schema (Friedman et al., 1999). Some attributes for the instances can be assigned values, other attributes are uncertain and part of the PRM definition is a specification of the parents of a given attribute, and a specification for how to construct a conditional probability distribution for an attribute of an instance which depends on the values for its parents attributes (Pool et al., 2005). PRM is considered as a more expressive model in comparison to the Bayesian networks it expresses much more information (Friedman et al., 1999).

QM is the representation language for PRM provided by IET [1] and is based on frames. OWL-QM extends OWL with a number of PRM constructs to represent quiddity facets (properties of associations between properties and classes) and slot-chains (lists of reference slots that specify the relation between instances of some classes and the relationships by which they are related to the attributes of another). An OWL class *"ProbabilisticRelationship"* is defined to represent the causal relations by using three object properties: *"parentPR"*, *"childPR"* and *"slotListPR"*. In addition to that, a set of vocabularies are used to define the probability distributions or tables in a probabilistic relationship. In fact a probability distribution is defined by creating an instance of *"ConditionalProbabilityTable"* in which the probability values will be stored. The values in a conditional probability are contained in instances of *"CPTCell"*, each of which is linked to a ConditionalProbabilityTable via the cptCell property.

### Fukushige's approach

The work of Fukushige (Fukushige, 2005) proposed a vocabulary for representing probabilistic knowledge in a RDF graph and a corresponding probability calculation framework using RDF and Bayesian network.

This framework distinguishes three kinds of probabilistic information encoded in RDF: probabilistic distribution (conditional and unconditional probabilities), observations (observed data) and probabilistic beliefs (posterior). The aim of this work is to describe probabilistic relations in a way that is both semantic web compatible and easy to map to a Bayesian network.

In (Fukushige, 2004), the framework for probability calculation is presented. In fact

[1] Information Extraction and Transport is a service company specialized in Bayesian Networks solutions

the different steps are summarized in the following:

- Describe the problem by a RDF graph using the proposed vocabulary.
- Convert the graph into a Bayesian network, and export it to a Bayesian network store.
- Describe the observation by a RDF graph.
- Convert the observation graph into a query for a Bayesian reasoner on the store and hand it to the reasoner.
- Do the calculations on the reasoner and import the result back to the RDF store and merge.

### 3.4.4 Comparison

This section presents a comparison between the two frameworks BayesOWL and Ontobayes which are the basis for our work.

#### Similarities

The two models BayesOWL and OntoBayes presented earlier have some points in common in fact both of them use the probability theory to express uncertainty information and model it in a Bayesian network. The two models translate an owl ontology into a Bayesian network and add classes to the owl file to annotate it with uncertainty.

#### Differences

The table above summarizes the major differences between the two frameworks based on the following criterion (random variables, nature of random variables, dependency relations and ontology engineering tasks).

Table 3.6: Major differences between BayesOWL and OntoBayes

| Criterion | BayesOWL | OntoBayes |
|---|---|---|
| Random Variables | Unable to represent multivalued random variables. The considered random variables are binary one either true or false | Represent multivalued random variables that are discrete |
| Nature of random variables | Classes | Object property or datatype. property. |
| Dependency relations | Use a set-theoretic approach based rules to model the dependency relations between variables. Domain specific method is proposed. | A property element <rdfs:dependsOn> is added to markup dependency information. A common dependency modeling method is proposed. |
| Ontology engineering tasks | This model focused on dealing with uncertainty in the three main tasks: domain modeling, ontology reasoning, and concept mapping between ontologies. | This model focused only on including uncertainty in domain modeling (knowledge representation). |

## 3.5 Fuzzy Web Ontology Languages

Besides the probabilistic approaches presented above, there are also other researchers who adopt the fuzzy sets theory as an uncertainty approach in ontology representation and reasoning. A brief description of Fuzzy OWL (Stoilos et al., 2005) and FOWL (Gao and Liu, 2005) will be presented in the following.

### Fuzzy OWL

Fuzzy OWL (f-OWL) is a method proposed by (Stoilos et al., 2005) for extending OWL with fuzzy sets theory in order to represent and reason with uncertainty information in the semantic web. The fuzzy extension of OWL DL focuses on OWL facts by adding degrees. For that purpose the f-OWL introduces an additional element < *owlx:degree*> to express the fuzziness degree added to the facts.

Once the uncertain information is represented, a reasoning task must be provided for f-OWL which will be realized through the combination of the syntactical extensions with the f-SHOIN. The f-SHOIN extends SHOIN to the fuzzy case by letting concepts and roles denote fuzzy sets of individuals and relations among them respectively. In f-SHOIN, the fuzzy knowledge base contains:
- Fuzzy TBox: a finite set of fuzzy concept axioms.

- Fuzzy RBox: a finite set of fuzzy role axioms.
- Fuzzy ABox: a finite set of fuzzy assertions.

**FOWL**

The work presented in (Gao and Liu, 2005) extends OWL language by encoding fuzzy constructors, axioms and constraints in order to map them to fuzzy DL. The extended OWL can represent fuzzy ontology as well as resolving fuzzy inference questions by constraint propagation calculus. In addition to the vocabularies, the authors present some rules for translating OWL to FOWL because from the viewpoint of fuzzy set, some common OWL concepts are also special fuzzy concepts.

## 3.6 Conclusion

In this chapter, we recalled the different models used for handling uncertainty and a special concern was devoted to the probabilistic graphical models with a whole description of the Bayesian networks. The major part of this chapter focused on the presentation of the different researches for handling uncertainty in ontology representation and reasoning based on two different approaches: probabilistic one and the fuzzy sets theory. These two approaches fail on representing ignorance which is resolved by applying the Dempster-Shafer theory. The concepts related to this theory and the different issues that motivate us to propose the development of our tool will be presented in the next chapter.

# 4

# BeliefOWL: An Evidential Extension to OWL

## 4.1 Introduction

To represent a domain of discourse correctly, uncertainty aspect should be taken into account since the real life is full of inconsistent, inaccurate and uncertain information. Most of the ongoing research in the field of applying uncertain representation and reasoning to the semantic web focuses on fuzzy logic and probability approaches as presented in the previous chapter. However not all the problems of uncertainty lend themselves to one of these approaches and the development of Bayesian networks as graphical models to represent this uncertainty can not handle with the situations where we are called to represent ignorance which can be resolved by applying the Dempster-Shafer theory (Shafer, 1976).

Due to the advantages of the Dempster-Shafer theory, we intend in this chapter to present our framework BeliefOWL as a way for representing uncertainty in the OWL ontology and propagating beliefs based on directed evidential network (Ben Yaghlane, 2002). We aim at this chapter to present the Dempster-Shafer theory as an approach for handling uncertainty in the semantic web as well as to describe the graphical structure of the directed evidential network as a model to represent knowledge under uncertainty by the use of the belief functions. Without omitting that a whole description of our framework's architecture will be presented. We will give in detail the different steps that have lead us to construct our framework as well as the algorithms that we traced.

Before presenting the BeliefOWL, we will expose in section 4.2 the necessary background material related to the Dempster-Shafer theory its preliminaries as well as a brief introduction to the directed evidential network as a graphical model for representing uncertain knowledge and propagating beliefs. Section 4.3 will be devoted to the enumeration of the different observations that motivate us to investigate in the ontology representation and reasoning. Our framework BeliefOWL will be described in detail in section 4.4 by presenting the different steps leading to it as well as the BeliefOWL algorithm which will be given in section 4.5. Finally, an example illustrating the different steps of our framework will be viewed in section 4.6.

## 4.2 Evidential Modeling and Reasoning

### 4.2.1 Belief Function Theory Preliminaries

Belief function theory, also known as Dempster-Shafer theory and theory of evidence, has been proposed for modeling someone's degrees of belief (Shafer, 1976). This theory provides a way to assign a belief to a single element of a set of values in the purpose to model uncertain information and to assure reasoning. In addition, this theory allows to represent uncertainty for knowledge representation because beliefs can be assessed to a set of hypothesis rather than to each one. As opposed to the classical probabilistic theories, the theory of evidence can represent the different levels of abstraction making it possible to distinguish between uncertainty and ignorance. Finally, the Dempster's combination rule provides an interesting method to combine the effect of different evidences to establish new belief.

In order to model uncertainty using the belief function theory, some used concepts have to be defined.

**Definition 4.1** ***Basic Belief Assignment.*** Let $\Theta$ be a finite set of elementary hypotheses, called *the frame of discernment*. We denote by $2^{\Theta}$ the set of all the subsets of $\Theta$.
A *basic belief assignment(bba)* is a function defined by :

$$\begin{aligned} &(i) \quad m : 2^{\Theta} \rightarrow [0,1], \\ &(ii) \quad m(\emptyset) = 0, \\ &(iii) \quad \sum_{A \subseteq \Theta} m(A) = 1. \end{aligned}$$

The value *m(A)* quantifies the part of belief that supports a subset of hypotheses. *m(A)* cannot support any more specific hypothesis by lack of information.

A *focal element* of m is defined as the subsets A of the frame of discernment $\Theta$ such that $m(A) \neq 0$.

**Definition 4.2** ***Belief Function.*** The belief function *bel* is defined as follows:

$$\begin{aligned} &(i) \quad bel : 2^{\Theta} \rightarrow [0,1], \\ &(ii) \quad bel(A) = \sum_{B \subseteq A} m(B), \forall A \subseteq \Theta. \end{aligned}$$

The value *bel(A)* expresses the total belief fully allocated to the subsets A of $\Theta$.

**Definition 4.3** ***Plausibility Function.*** The plausibility function *pl* is defined as follows*:*

$$(i)\ \ pl : 2^{\Theta} \rightarrow [0,1],$$
$$(ii)\ pl(A) = \sum_{B \bigcap A \neq \emptyset} m(B), \forall A \subseteq \Theta.$$

The value *pl(A)* expresses the maximum amount of belief that might support the subset A.

**Definition 4.4** ***Combination.***

The $\oplus$ symbol represents *Dempster's rule of combination* in its normalized form and $\textcircled{\cap}$ represents the *conjunctive combination*, i.e., the same operation as Dempster's rule of combination except the normalization is not performed.

The Dempster's rule of combination (as well as its conjunctive form) is applicable to combine belief mass functions that are on the same $\Theta$. Given $m_1$ and $m_2$, the rule can be written as: $\forall A \subseteq \Theta$,

$$m_{1 \oplus 2}(A) = \frac{\sum_{B \bigcap C = A} m_1(B) m_2(C)}{1 - \sum_{B \bigcap C = \emptyset} m_1(B) m_2(C)}$$

**Definition 4.5** ***Generalized Bayesian Theorem (GBT).***

Suppose we have two distinct variables X and Y defined on the spaces $\Theta_X$ and $\Theta_Y$, respectively. Given that X is the parent of Y, let $bel^Y[x_i]$ be the additional information representing conditional belief function induced on the space $\Theta_Y$ given $x_i$ element of $\Theta_X$.

If we want to compute $bel^X[y](x)$ for any x $\subseteq \Theta_X$ and y $\subseteq \Theta_Y$, we derive this belief from the *generalized bayesian Theorem*(GBT).

For given belief function bel, (Smets, 1993) defines a function

$$\text{b: } 2^{\Theta} \longrightarrow [0,1]$$
$$\text{b(A)} = \text{bel(A)} + \text{m}(\emptyset)$$

For any any x $\subseteq \Theta_X$ and y $\subseteq \Theta_Y$, the GBT permits to build the belief function $bel^X[y](x)$ by:

$$\text{bel}^X[y](x) = b^X[y](x) - b^X[y](\emptyset)$$
$$b^X[y](x) = \prod_{x_i \in \bar{x}} b^Y[x_i](\bar{y})$$

**Definition 4.6** ***Disjunctive Rule of Combination (DRC).***

*The disjunctive rule of combination* (DRC) is used to compute $bel^Y[x](y)$ for any x $\subseteq\Theta_X$ and y $\subseteq\Theta_Y$. Indeed, the DRC permits to build the belief function $\text{bel}^Y[x](y)by$ :

$$\text{bel}^Y[x](y) = b^Y[x](y) - b^Y[x](\emptyset)$$
$$b^Y[x](y) = \prod_{x_i \in \bar{x}} b^Y[x_i](y)$$

### 4.2.2 Directed Evidential Network

In the following, we present the directed evidential network with conditional belief functions (DEVN) (Ben Yaghlane et al., 2003) which is a generalization of the evidential networks described in (Xu, 1995).

#### Graphical structure

The directed evidential networks with belief functions (DEVN) are models introduced by (Ben Yaghlane et al., 2003) to represent knowledge under uncertainty by using the belief functions.

To define this network, one can distinguish between two levels:

- Qualitative level: The DEVN is a directed acyclic graph (DAG) where the nodes represent variables and the directed arcs link between nodes and describe conditional dependence relations between these variables.
- Quantitative level: The relations are expressed by conditional belief functions for each variable given its parents.

Each variable X in this network has a set of possible values, called *frame of discernment*, and the parents of X are denoted by $Pa_X$.

Two kinds of belief functions are depicted to represent uncertainty in the DEVN: the *prior belief function* and the *conditional belief function*. The former denoted as $bel_X^0$ concerns the root node where ($Pa_X \neq \emptyset$) and the latter denoted by $bel^X[Pa_X]$ expresses the belief function of a node X given the value taken by its parents.

#### Inference

The propagation algorithm to be applied depends on the nature of the network we are working with.

- **Propagation algorithm for singly-connected evidential network (Ben Yaghlane and Mellouli, 1999)**:
  The key idea of this algorithm is inspired from Pearl's algorithm (Pearl, 1988). It consists on two steps:

  - *Initialization Process*: In this step, the a priori belief functions of each variable representing a node of the network is computed using propagation method.
  - *Updating Process*: This step is performed when a new observation is introduced at a given node. In fact, arriving at a node X, the algorithm computes all the incoming messages as well as a calculation of $\pi_X$ value, $\lambda_X$ value, new marginal $bel^X$ and all outgoing messages is done.

- **Propagation algorithm for multiply-connected evidential network** :
  In (Ben Yaghlane, 2002), the authors proposed a computational data structure *Modified Binary Join Tree* to maintain the (in)dependence relations of the original directed evidential network with conditional belief functions. The propagation process is performed by using the generalized Bayesian theorem (GBT) and the disjunctive rule of combination (DRC).

## 4.3 Motivation and Overview of BeliefOWL

Based on the background material discussed in the previous chapters, we depict in this section the main motivations that encourage us to propose BeliefOWL as a framework for ontology representation and reasoning under uncertainty. In fact:

- Modeling uncertainty in ontology engineering tasks (in our case ontology representation and ontology reasoning) is necessary in nowadays research because it allows to overcome the difficulties arising from the crisp logics of the OWL language which is not able to handle incomplete or partial knowledge. Taking into account the uncertainty aspect in ontology representation permits to strengthen the OWL with additional expressive power to quantify the uncertainty of classes and relations. In ontology reasoning, handling uncertainty enables to support inference tasks by using inference procedures to compute the degree of overlap or inclusion between two concepts.

- Belief function theory provides a well developed mathematical model to deal with uncertainty. It is a suitable theory to express the total ignorance or the partial one about information concerning classes and relations within an ontology.

- The directed graphs and particularly the directed evidential network are effective and more appropriate graphical representations for uncertainty knowledge. In addition to that, the use of conditional belief functions provides a more natural representation of the uncertainty in the relationships among the variables of a graph.

In the following sections, we will present our framework BeliefOWL in detail, describing the different steps to be followed to contribute to our tool.

## 4.4 Presentation of BeliefOWL

We resume in this figure the different steps followed leading to our framework.

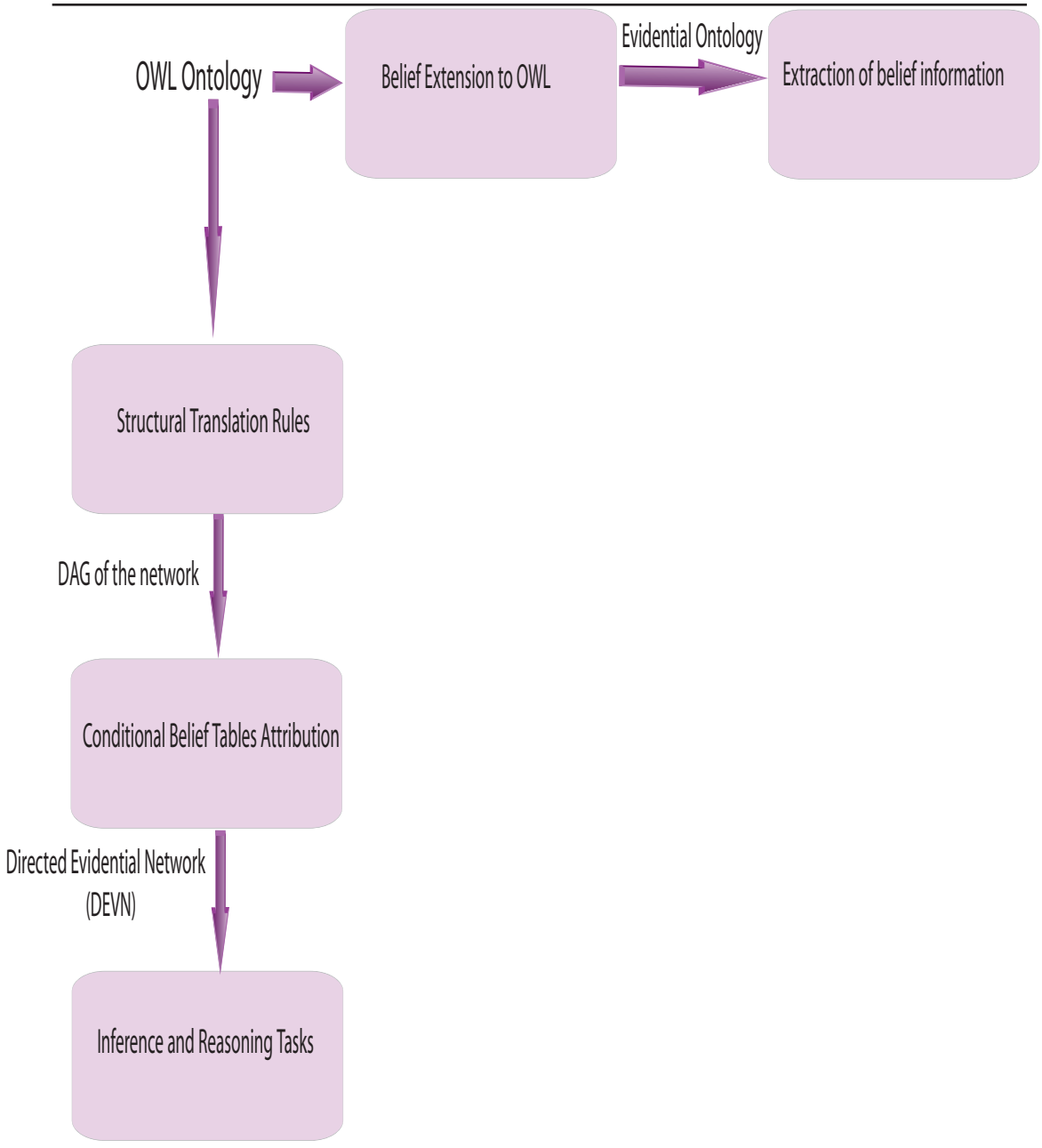


Figure 4.1: BeliefOWL Framework

As shown in the figure 4.1, the BeliefOWL has as input an OWL ontology and as output an evidential network. To obtain this network, we will follow some steps that will be described in detail through this chapter.

In *Belief extension to OWL*, we will focus on how to make an ontology, expressed on crisp logics, able to represent uncertain information, a partial ignorance or a total one by including belief masses. Once the evidential information has been included, it will be used later for the construction of the conditional beliefs of every evidential network's nodes. This network will be get through a conversion of our OWL ontology by applying some *structural translation rules* and then *conditional belief tables* will be constructed. Finally some *reasoning tasks* will be held across the directed evidential network.

### 4.4.1 A Belief Extension to OWL

In chapter 2, we presented that an OWL ontology can define classes (named or anonymous), properties (object properties, datatype properties) and individuals (instances of a class). In this master we will focus only on including uncertain information for classes. In other words, the belief masses will be attributed to the different classes of an OWL ontology.

For this purpose, we have to define some new classes able to represent and to introduce this uncertain information. Two types of evidence are encoded into the original ontology: prior evidence and conditional evidence. These evidences, expressed in the form of mass functions, are provided by domain experts.

- **Prior evidence**: We define two classes to express the prior evidence <*beliefDistribution*> and<*priorBelief*>. The former is used to enumerate the different masses of the elements of the frame of discernment of a variable. It has an object property <*hasPriorBelief*> that specifies the relation between classes <*beliefDistribution*> and <*priorBelief*> indicating that the instances of <*priorBelief*> are elements of one instance of <beliefDistribution>. The latter expresses the prior evidence and has a datatype property <*massValue*> which enables to assign a mass value between 0 and 1 to an element of the frame or to a set of elements.

  **Example:**
  Let X and Y be two variables formalizing two classes (sunbathing and relaxation where the former is a subclass of the latter one) of a given ontology. These variables can have two possible values either true or false.

  Let $\theta_X = \{T_s, F_s\}$ be the frame for X and $\theta_Y = \{T_r, F_r\}$ be the frame for Y. In order to represent the prior evidence, let $bel_0^X$ and $bel_0^Y$ be two prior belief functions over $\theta_X$ and $\theta_Y$, respectively and be given by their respective mass functions $m_0^X$ and $m_0^Y$. We suppose that we have the following masses related to the variable Y. $m_0^Y\ (\{\mathrm{T}_r\}) = 0.5, \quad \mathrm{m}_0^Y\ (\{\mathrm{F}_r\}) = 0.4, \quad \mathrm{m}_0^Y\ (\{\theta_Y\}) = 0.1$

  The following table encodes how the prior evidence was integrated.

```
<beliefDistribution rdf:ID= "bel(Y)">
    <hasPriorBelief rdf:resource = "# m(T_r) />
    <hasPriorBelief rdf:resource = "# m(F_r) />
    <hasPriorBelief rdf:resource = "# m(θ_Y) />
</beliefDistribution>
<PriorBelief rdf:ID= "m(T_r)">
    <massValue>0.5</massValue>
</PriorBelief>
<PriorBelief rdf:ID= "m(F_r)">
    <massValue>0.4</massValue>
</PriorBelief>
<PriorBelief rdf:ID= "m(θ_Y)">
    <massValue>0.1</massValue>
</PriorBelief>
```

- **Conditional evidence**: It's defined through two main classes <*beliefDistribution*> and<*condBelief*>. The former is the same as in the case of prior evidence but has an object property <*hasCondBelief*> indicating the relation between classes<*beliefDistribution*> and <*CondBelief*>. The latter identifies the conditional evidence and has a datatype property<*massValue*>.

  **Example:**
  Let's consider the following conditional beliefs. We adopt the notation m[X](Y) to represent the conditional mass m(Y | X).
  $m[\{T_s\}](\{T_r\}) = 0.6$, $m[\{T_s\}](\{\theta_Y\}) = 0.4$, $m[\{F_s\}](\{F_r\}) = 0.92$, $m[\{T_s\}](\{\theta_Y\}) = 0.08$.

  The corresponding integration is represented by the following table.

```
<beliefDistribution rdf:ID= "bel[X](Y)">
    <hasCondBelief rdf:resource = "# m[{T_s}]({T_r}) />
    <hasCondBelief rdf:resource = "# m[{T_s}]({θ_Y}) />
    <hasCondBelief rdf:resource = "# m[{F_s}]({F_r}) />
    <hasCondBelief rdf:resource = "# m[{T_s}]({θ_Y}) />
</beliefDistribution>
<condBelief rdf:ID= "m[{T_s}]({T_r})">
    <massValue>0.6</massValue>
</condBelief>
<condBelief rdf:ID= "m[{T_s}]({θ_Y})">
    <massValue>0.4</massValue>
</condBelief>
<condBelief rdf:ID= "m[{F_s}]({F_r})">
    <massValue>0.92</massValue>
</condBelief>
<condBelief rdf:ID= "m[{T_s}]({θ_Y})">
    <massValue>0.08</massValue>
</condBelief>
```

Through this integration step, we augmented the ontology with prior belief masses and conditional belief masses, we obtained an evidential ontology. This extension of OWL is not enough to get an evidential network, some other steps should be followed to be able to construct the different nodes of the network and the dependency relations among them. It will be the main focus of the next subsection.

### 4.4.2 Constructing an Evidential Network

As mentioned before, we distinguish in an evidential network two levels: a qualitative level and a quantitative one. In this subsection, we are interested on the qualitative one: on how to construct the DAG of the belief network from an OWL ontology by specifying the different kinds of nodes that have to be created in addition to the definitions of the relations between these nodes.

To handle that we will specify the OWL statements to be converted to a node. In fact we are interested in classes, in some class axioms and boolean combinations that may exist between the different classes.

The classes are introduced by <owl:class>, are described through class axioms (<owl:equivalentClass>,<rdfs:subclassOf>, <owl:disjointWith>) and a set of boolean combinations of classes exists to specify OWL logical operators describing the nature of the relations between classes such as (<owl:unionOf>, <owl:intersectonOf>).

We will focus on each of these OWL statements showing how the nodes are constructed and how the arcs are created.

We have to mention that all the examples given below are extracted from the ontology travel.owl[1] which is an ontology for a semantic web of tourism.

***OWL statement <owl:class>***
A named class described by a class identifier "rdfs:ID" is mapped into a variable node in the translated evidential network. This node is represented via the different mass functions related to that variable.

**Example 1:** A named class "Accommodation" defined as "<owl:class rdf:ID="Accommodation"/> is represented as:


Figure 4.2: Single Class

[1] http://protege.cim3.net/file/pub/ontologies/travel/travel.owl

***OWL statement <rdfs:subClassOf>***
When two classes are related to each other by a constructor <rdfs:subClassOf>, a directed arc is drawn from a superclass node to the child subclass node.

**Example 2:** The following example shows that the class "Safari" is a subclass of the two classes "Adventure" and "Sightseeing".

```
<owl:Class rdf:ID= "Safari">
  <rdfs:subClassOf rdf:resource="#Adventure"/ >
  <rdfs:subClassOf rdf:resource="#Sightseeing"/ >
</owl:Class>
```

This statement can be mapped into a subgraph in the translated evidential network as shown in the following figure:

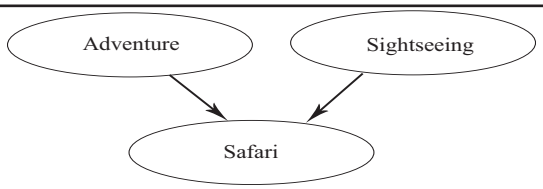


Figure 4.3: "rdfs:subClassOf"

***OWL statement <owl:disjointWith>***
When two classes are related to each other by a disjoint relation (owl:disjointWith), a new node is created in the translated evidential network and a directed arc is drawn between the two classes and the node added.

**Example 3:** The following example shows that the classes "Adventure" and "Relaxation" are disjoint with the class "Sports".

```
<owl:Class rdf:ID= "Sports">
  <owl:disjointWith>
    <owl:class rdf:about="#Adventure"/ >
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:class rdf:about="#Relaxation"/ >
  </owl:disjointWith>
</owl:Class>
```

The following example shows the representation of this statement in the subgraph of the evidential network.

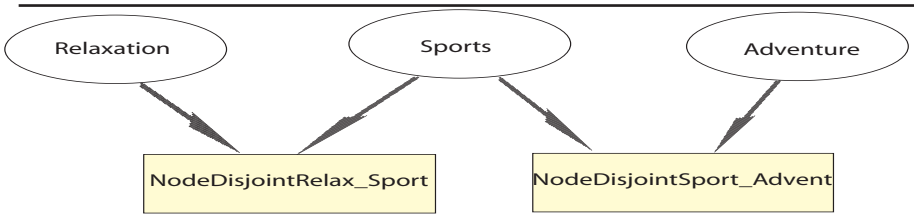


Figure 4.4: "owl:disjointWith"

***OWL statement <owl:equivalentClass>***
When two classes are equivalent to each other by a (owl:equivalentClass) statement, a new node is created in the translated evidential network and a directed arc is drawn between the two classes and the node added.

**Example 4:** The following example shows that the class "QuietDestination" and "FamilyDestination" are equivalent to each other.

```
<owl:Class rdf:ID= "QuietDestination">
  <owl:equivalentClass rdf:resource="#FamilyDestination"/ >
</owl:Class>
```

The following example shows the representation of this statement in the subgraph of the evidential network.

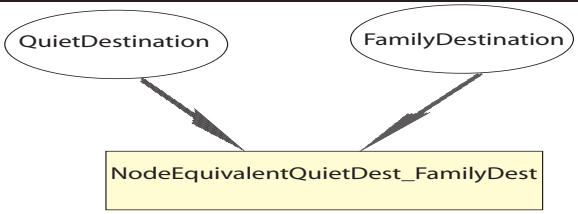


Figure 4.5: "owl:equivalentClass"

***OWL statement <owl:intersectionOf>***
A class C may be defined as the intersection of some classes $C_i$(i,. . . ,n) through <owl:intertionOf>. This statement can be represented in the translated evidential network by an arc from each $C_i$ to C and another one from C and each $C_i$ to a new node created for representing the intersection.

**Example 5:** The example above shows that the class "Booking" is the intersection of the two classes "BudgetAccommodation" and "Hotel".

```
<owl:Class rdf:ID= "Booking">
  <owl:intersectionOf rdf:parseType = "Collection">
    <owl:class rdf:about="#BudgetAccommodation"/ >
    <owl:class rdf:about="#Hotel"/ >
  </owl:intersectionOf>
</owl:Class>
```

The statement <owl:intersectionOf > can be mapped into a subgraph in the translated evidential network as shown in the following figure:

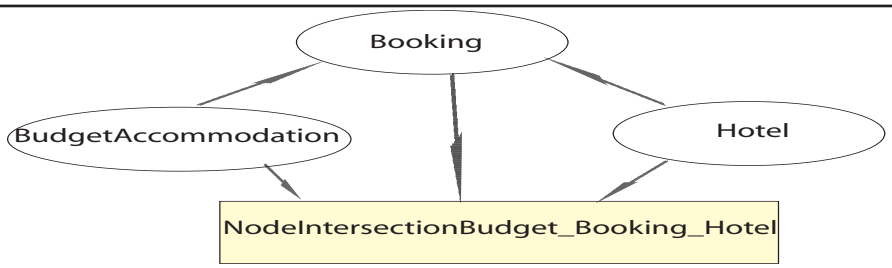


Figure 4.6: "owl:intersectionOf"

***OWL statement <owl:unionOf>***

A class C may be defined as the union of some classes $C_i$(i,. . . ,n) through <owl:unionOf>. This statement can be represented in the translated evidential network by an arc from C to each $C_i$ to C and another one from C and each $C_i$ to a new node created for representing the union.

**Example 6:** The following example shows that the class "DestiantionOffers" is the union of the two classes "Sports" and "Adventure".

```
<owl:Class rdf:ID= "DestinationOffers">
  <owl:unionOf rdf:parseType = "Collection">
    <owl:class rdf:about="#Sports"/ >
    <owl:class rdf:about="#Adventure"/ >
  </owl:unionOf>
</owl:Class>
```

The statement <owl:unionOf > can be mapped into a subgraph in the translated evidential network as shown in this figure:

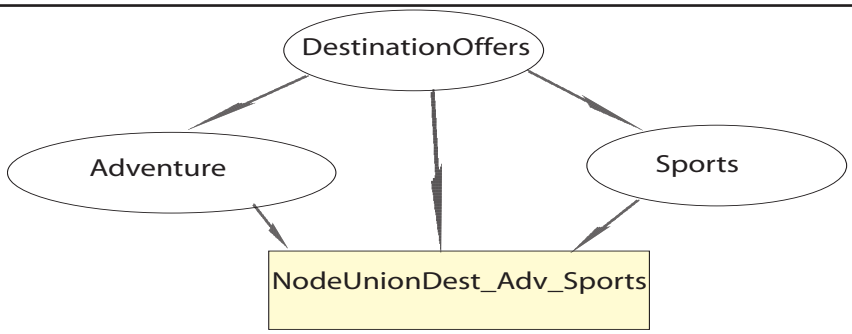


Figure 4.7: "owl:unionOf"

At the end of this step and by the application of the previous rules, the network is constructed: the nodes are specified and the different arcs between the nodes are drawn.

The obtained network is a directed acyclic graph even by adding the nodes expressing the logical operators between classes, the cycle is not formed. Only the use of the constructor "rdfs:subClassOf" can lead to a cycle. In fact, in an ontology two classes can be subclasses of each other (a class D is a subclass of a class E and E is a subclass of D) and by the application of the rule related to this OWL statement we can get a cycle.

To avoid this situation, the two classes (for example E and D) can be considered as equivalent and thus we apply the rule related to the "owl:equivalentClass".

In addition to that all arcs in the network are directed: when two classes are related to each other by a class axiom or by a logical operator, an arc is drawn between the different nodes and when they are not related by an axiom, the corresponding nodes are d-separated with each other.

### 4.4.3 Evidence Attribution

In the previous step, and by applying the structural translation rules, we constructed the DAG of our network. To complete the translation, the remaining issue is to assign masses for each node of the network. Considering the DAG that we have got, we can depict two kinds of nodes:

- **ClassesNodes**: We define classesNodes as the nodes representing the different classes of our taxonomy and defined by <owl:class>.

- **ConstNodes**: We mean by these nodes those related to the constructors of our taxonomy (owl:intersectionOf, owl:disjointWith, owl:equivalentClass and owl:unionOf) without considering <rdfs:subClassOf> because this kind of constructor is not represented by a specific node.

Masses distribution depends on the kind of nodes we are working with. In fact for the classesNodes, we attribute the masses given into the evidential ontology which is created after the extension of the OWL ontology with belief information. The prior belief functions and the conditional ones will be extracted from this evidential ontology to be assigned to the corresponding nodes. Concerning the constNodes, masses will be attributed according to the constructor we are talking about. In fact if we have a node created to depict an intersection between two classes, the mass will be attributed by applying the Dempster's rule of combination. Concerning the node representing an union, the disjunctive rule of combination will be applied in that case.

Once our evidential network is constructed and the masses are assigned to each node a propagation process can be performed. It's the objective of the next section.

### 4.4.4 Inference in the Network

One important task to do across a network is to perform a propagation process. In most of the cases, the network obtained is a multiply-directed evidential network. A propagation process has been proposed in (Ben Yaghlane, 2002) and an algorithm has been applied in (Trabelsi, 2007).

This algorithm consists on transforming the initial directed evidential network into a modified binary join tree. In order to make inference efficiently in this graphical structure, two rules are proposed to be applied: the disjunctive rule of combination (DRC) and the generalized Bayesian Theorem (GBT).

## 4.5 BeliefOWL Algorithm

In this section, we present our algorithm for the BeliefOWL as a tool proposed to translate an OWL ontology into an evidential network after including mass functions.

### 4.5.1 Algorithm Overview

The BeliefOWL algorithm follows five major steps leading to a translation of an OWL ontology into a belief network. This is resumed by the following figure:

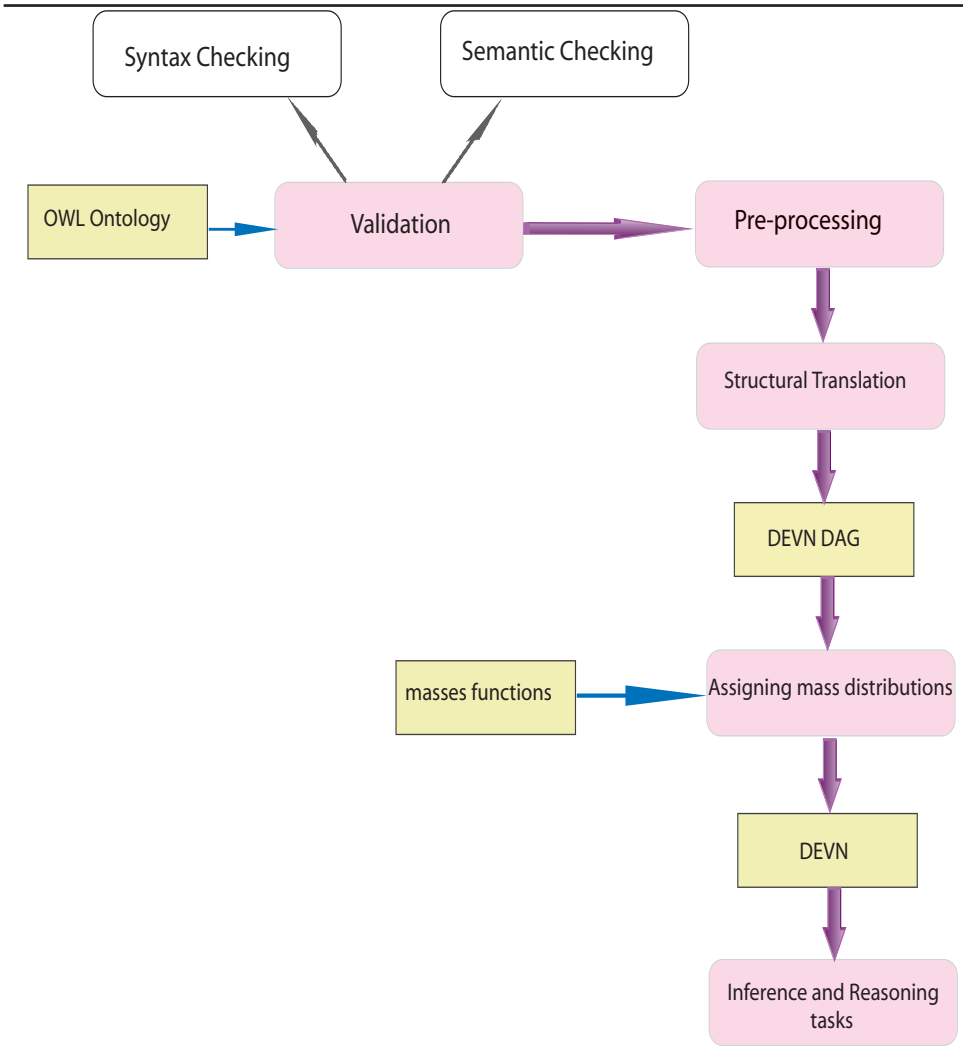


Figure 4.8: BeliefOWL Algorithm

1. The validation step: It is an important step which verifies that the given ontology is syntactically correct and semantically consistent.

2. The preprocessing step: Once we are sure that the ontology is valid, some major information concerning the ontology's classes should be collected. We will be interested in mentioning for each class its superclasses, its subclasses, the classes that are disjoint with, or equivalent to that class, as well as the classes that are described through logical operations (intersection and union).

3. Establishing structural translation rules: At this stage the ontology will be translated to an evidential network by applying the different rules mentioned in section 4.4.2.

4. Assigning mass distributions: At this stage we are interested on the quantitative level so mass distribution functions will be assigned to the nodes of the created DAG.

5. Belief propagation and reasoning tasks: We propagate uncertainties through the evidential network and do some reasoning tasks such as concept satisfiability and concept subsumption.

In this dissertation, the validation step will not be detailed because it is an easy step and it can be done through APIs available to check the syntax and the semantics of an ontology. But we are interested on the next steps which will be described in detail.

### 4.5.2 Preprocessing Step

This step is mainly devoted to collect information of the different classes of an ontology. This collection helps later on to construct the DAG of the network. For that purpose, this step is very important and consists on specifying for each class of the ontology its superclasses, its subclasses, the classes that are equivalent to it as well as those that are disjoint with it without omitting those obtained through an intersection or an union.

This step requires as input the *path* of the ontology $O$ which has been tested to be correct syntactically and semantically consistent. Once the ontology is read, a task of (*describeClasses algorithm*) is performed. We will not detail this algorithm because it is realized through an available API.

**Algorithm** PRE-PROCESSING.

1. **If** $path_1 \neq$ " " and test **Then**
   Read(O)
   **End If**
2. DescribeClasses(O).

### 4.5.3 Structural Translation Step

This step has as input the information collected for each class in the previous step (*the preprocessing*) and the rules described in section 4.4.2. Based on this input, two procedures will be executed: *createNode* and *addLink* which consist respectively on creating the nodes of the DAG and adding the links between them leading to the generation of the DAG of the evidential network.

**Algorithm** *Structural Translation.*

**For** each cls **do**
**If** cls is a named class **then**
createNode(cls)
**elseIf** cls is an anonymous class **then**
**If** cls.hasSubClasses = true **then**
createNode(cls)
**for** each $cls_i \in listSubClasses$
createNode($cls_i$)
addLink(cls, $cls_i$)
**End For**
**elseIf** cls.isDisjointWith = true **then**
createNode(cls)
**for** each $cls_i \in listDisjointClasses$
createNode($cls_i$)
createNode(nodeDisjoint)
addLink(cls, nodeDisjoint)
addLink($cls_i$, nodeDisjoint)
**End For**
**elseIf** cls.isEquivalent = true **then**
createNode(cls)
**for** each $cls_i \in listEquivalentClasses$
createNode($cls_i$)
createNode(nodeEquivalent)
addLink(cls, nodeEquivalent)
addLink($cls_i$, nodeEquivalent)
**End For**
**elseIf** cls.isIntersectionClass = true **then**
createNode(cls)
createNode(nodeIntersection)
addLink(cls, nodeIntersection)
**for** each $cls_i \in listIntersectionClasses$
createNode($cls_i$)
addLink($cls_i$, cls)
addLink($cls_i$, nodeIntersection)
**End For**
**elseIf** cls.isUnionClass = true **then**
createNode(cls)
createNode(nodeUnion)
addLink(cls, nodeUnion)
**for** each $cls_i \in listUnionClasses$
createNode($cls_i$)
addLink(cls, $cls_i$)
addLink($cls_i$, nodeIntersection)
**End For**
**End If**

**End For**

### 4.5.4 Assigning Mass Distributions

The masses are provided by an expert. These masses are given in the evidential ontology. So, once we get the DAG of the network, the masses are attributed as follows. Concerning the classesNodes, we extract the masses from the ontology and they are assigned to the corresponding node: If it is a root node then a prior mass will be attributed, if it is another node belonging to the set of classNodes then a conditional belief is assigned. Concerning the constNodes the masses attribution depends on the logical relation it holds. If it is an intersection a Dempster's rule of combination will be applied and if it is an union the disjunctive rule of combination will be computed.

### 4.5.5 Inference Step

To do the propagation of beliefs in the multiply-connected evidential network, we will use the BeliefNet tool proposed in (Trabelsi, 2007) which allows belief propagation in this kind of network.

## 4.6 Application

Throughout this dissertation, we used examples extracted from the travel.owl ontology. In this section we choose to use a simple taxonomy ontology, nature.owl, because we are only interested in some specific constructors. From this ontology, we will illustrate the different steps of our tool BeliefOWL.

```
<? xml version ="1.0">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:owl="http://www.w3.org/2002/07/owl"
         xmlns ="http://www.cs.umbc.edu/zding1/owl/nature.owl" >
<owl:Ontology rdf:about ="http://www.cs.umbc.edu/zding1/owl/nature.owl">
  <owl:versionInfo">v1.0</owl:versionInfo>
<owl:Class rdf:ID="Animal">
<owl:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
<owl:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <owl:disjointWith rdf:resource="#Male"/>
</owl:Class>
<owl:Class rdf:ID="Human">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
<owl:Class rdf:ID="Man">
  <owl:intersectionOf rdf:parseType="Collection"/>
    <owl:Class rdf:resource="#Human"/>
    <owl:Class rdf:resource="#Male"/>
 </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:ID="Woman">
  <owl:intersectionOf rdf:parseType="Collection"/>
    <owl:Class rdf:resource="#Human"/>
    <owl:Class rdf:resource="#Female"/>
 </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:ID="#Human">
  <owl:unionOf rdf:parseType="Collection"/>
    <owl:Class rdf:resource="#Man"/>
    <owl:Class rdf:resource="#Woman"/>
 </owl:unionOf>
</owl:Class>
</rdf:RDF>
```

This taxonomy shows that we have six concepts: "Animal", "Male", "Female", "Human", "Woman" and "Man". These concepts are related to each other by different constructors. In fact:

- "Male", "Female" and "Human" are **subclasses** of the concept "Animal".
- The concept "Man" is the **intersection** of "Male" and "Human". It is the same case of "Woman" which is the **intersection** of "Female" and "Human".

- "Human" is the **union** of "Woman" and "Man".

- "Male" and "Female" are related to each other by a **disjoint** constructor.

Having a general overview of the taxonomy we can construct the DAG of our network by applying the set of structural translation rules. The figure above shows the different nodes of the DAG as well as the relations existing between them.

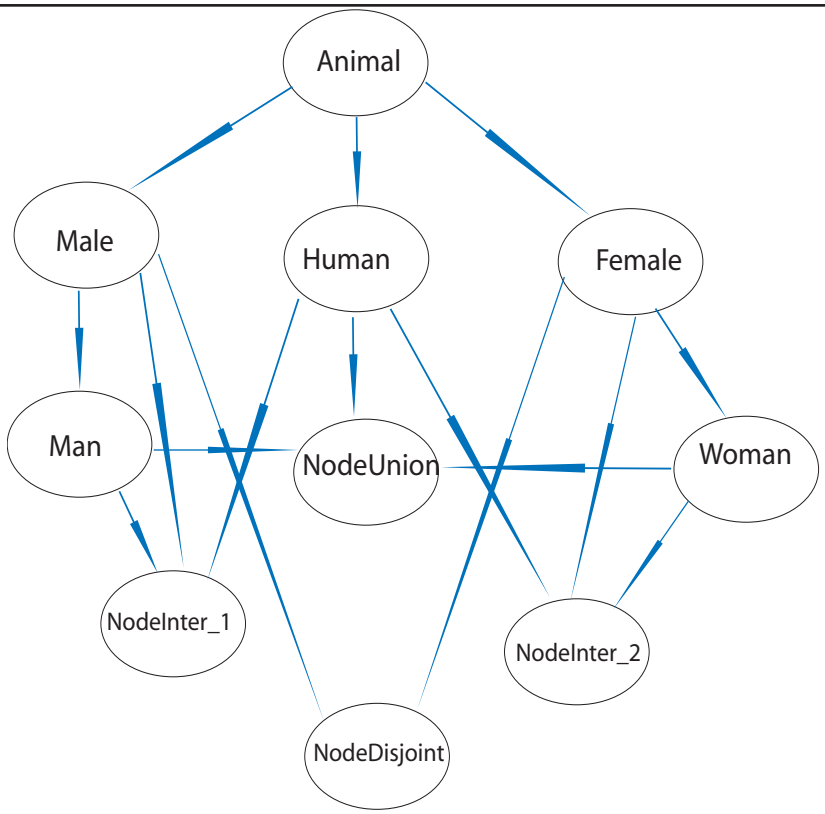


Figure 4.9: DAG of the evidential network.

Once the DAG of the evidential network is constructed, the corresponding a priori and conditional belief functions will be assigned to the different classNodes.

We adopt a simple notation of the different nodes as shown in the following table:
We suppose that each variable corresponding to a specific node can only have two states either true or false. We assign the following masses:

Table 4.1: Notation

| Node | Notation |
|---|---|
| Animal | A |
| Human | H |
| Male | Ml |
| Man | M |
| Woman | W |
| Female | F |

$$m(A) = \begin{matrix} a \\ \bar{a} \\ \theta_A \end{matrix} \begin{pmatrix} 0.4 \\ 0.5 \\ 0.1 \end{pmatrix} \qquad m[A](Ml) = \begin{matrix} & a & \bar{a} \\ ml & 0.5 & 0 \\ \bar{ml} & 0 & 0.6 \\ \theta_m l & 0.5 & 0.4 \end{matrix} \qquad m[A](H) = \begin{matrix} & a & \bar{a} \\ h & 0.1 & 0 \\ \bar{h} & 0 & 0.5 \\ \theta_h & 0.9 & 0.5 \end{matrix}$$

$$m[A](F) = \begin{matrix} & a & \bar{a} \\ f & 0.8 & 0 \\ \bar{f} & 0 & 0.6 \\ \theta_f & 0.2 & 0.4 \end{matrix} \qquad m[F](W) = \begin{matrix} & f & \bar{f} \\ w & 0.75 & 0 \\ \bar{w} & 0 & 0.5 \\ \theta_w & 0.25 & 0.5 \end{matrix} \qquad m[Ml](M) = \begin{matrix} & ml & \bar{ml} \\ m & 0.75 & 0 \\ \bar{m} & 0 & 0.5 \\ \theta_m & 0.25 & 0.5 \end{matrix}$$

## 4.7 Conclusion

In this chapter, we have presented a theoretical aspect of our BeliefOWL tool. This tool deals with uncertainty in ontology representation and ontology reasoning based on the belief functions theory. An algorithm have been traced to present the different steps leading to our tool. At the end of the chapter we presented an example illustrative showing the translation of an OWL ontology into a directed evidential network as well as a propagation of beliefs is performed by using the BeliefNet tool.

# 5
# Conclusion and future work

With the development of semantic web, ontologies have become used to represent a specific domain of discourse by capturing the knowledge about concepts and their relations. To facilitate information sharing, a formal language, OWL, is well used but it fails to represent partial information. Dealing with uncertainty in ontology engineering tasks (ontology modeling, ontology reasoning and ontology mapping) seems to be crucial. Many researchers took into account that it is very interesting to represent imprecise information or a vague one, they proposed approaches for combining the web ontology language OWL with probability theory or the fuzzy set but fewer are those who considered the advantages of the Dempster-Shafer theory for representing uncertain information in an ontology or to use a formalism such as the directed evidential network for propagation.

In this dissertation, we proposed an approach for representing uncertainty in semantic web field. We used the evidential theory for uncertainty representation and the directed evidential network as a graphical model for inferring about new knowledge. We described the architecture of our tool BeliefOWL able to translate an ontology into a directed evidential network by applying a set of rules in order to present every constructor and class of an ontology into a corresponding node. Once the network is constructed, further reasoning tasks can be done across it.

In our future work, we will tend to investigate different improvements to our tool. In this master thesis, we worked with a taxonomy. In fact we translated only the classes, the class axioms (rdfs:subClassOf, owl:equivalentClass and owl:disjointWith) as well as the logical relations among the concept classes (owl:unionOf and owl:intersectionOf). Further work can carry about the properties and individuals which are not represented in this work. The masses that are assigned to the different nodes are extracted from an evidential ontology where the prior beliefs are given by an expert, so this assignment can be done automatically in the future through a learning process.

This work can be ameliorated in order to be applied in real life domains such as banking, medicine,...

Finally, we hope that this master thesis can contribute to put forward research work on the uncertainty in ontology engineering tasks based on the belief functions theory.

# Bibliography


Antoniou, G. and Van Harmelen, F. (2004). Web ontology language: Owl. In *Handbook On Ontologies*, International Handbooks on Information Systems, Chapter 4. Springer-Verlag, pages 67–92.

Ben Yaghlane, B. (2002). Uncertainty representation and reasoning in directed evidential networks. PhD thesis, Institut Supérieur de Gestion de Tunis Tunisia.

Ben Yaghlane, B. and Mellouli, K. (1999). Updating directed belief networks. In Hunter, T. and Parsons, S., editors, *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty ECSQARU'99.* Lecture Notes in AI 1638, Springer-Verlag, pages 43- 54.

Ben Yaghlane, B., Mellouli, K., and Smets, P. (2003). Directed evidential network with conditional belief functions. In Nielsen, T. and Zhang, N., editors, *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (ECSQARU'2003) Springer-Verlag LNAI 2711, pages 291-305.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American* , volume 284(5), pages 34-43.

Borst, P., Akkermans, J. M., and Top, J. L. (1997). Engineering ontologies. *International Journal on Human Computer Studies*, volume 46(2/3), pages 365-406.

Bruyninckx, H. (2002). Bayesian probability. Course's notes, Dept. of Mechanical Engineering, K.U.Leuven, Belgium, http://people.mech.kuleuven.be/ bruyninc/pubs/urks.pdf.

Butz, C. and Fang, F. (2005). Incorporating evidence in bayesian networks with the select operator. In *Proceeding of the 18th Conference of the Canadian Society For Computational Studies of Intelligence (Canadian AI 2005), Advances in Artificial Intelligence,* Volume 3501/2005 of LNCS, Victoria, Canada, pages 297-301.

Choi, N., Song, I., and Han, H. (2006). A survey on ontology mapping. *SIGMOD Record*, volume 35(3), pages 34-41.

Costa, P., Laskey, K., and Laskey, K. (2005). PR-OWL: A Bayesian Ontology Language for the Semantic Web. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005) at the 4th International Semantic Web Conference (ISWC),*Galway, Ireland, pages 23-33.

Costa, P., Laskey, K., and Laskey, K. (2006). Probabilistic ontologies for efficient resource sharing in semantic web services. In *Proceedings of the second workshop on Uncertainty Reasoning for the Semantic Web (URSW 2006) at the 5th International Semantic Web Conference (ISWC),*Athens, GA, USA.

Ding, Z. (2005). BayesOWL: A Probabilistic Framework for Semantic Web. PhD thesis, University of Maryland, Baltimore County.

Ding, Z. and Peng, Y. (2004). A Probabilistic Extension to Ontology Language OWL. In *Proceedings of the* $37^{th}$ *Hawaii International Conference On System Sciences (HICSS'04)*, Big Island, Hawaii, Track4, volume 4.

Ding, Z., Peng, Y., and Pan, R. (2004). A Bayesian approach to uncertainty modeling in owl ontology. In *Proceedings of the International Conference On Advances In Intelligent Systems Theory And Applications.*

Ding, Z., Peng, Y., and Pan, R. (2005). BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. In *Soft Computing in Ontologies and Semantic Web*, Studies in Fuzziness and Soft Computing. Springer-Verlag.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Y. (2004). Ontology matching: A machine learning approach. In *Handbook of Ontologies*, International Handbooks on Information Systems, Chapter 18. Springer-Verlag, pages 385–404.

Dubois, D. and Prade, H. (1988). Possibility theory. Plenum Press, New-York.

Ehrig, M. and Staab, S. (2004). Qom - Quick Ontology Mapping. In *The Semantic Web Proceedings ISWC04.* Springer-Verlag LNCS 3298, pages 289–303.

Ehrig, M. and Sure, Y. (2004). Ontology Mapping - An Integrated Approach. *SIGMOD Record*, volume 35(3), pages 34-41.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99),* San Francisco, CA, USA, 1999. Morgan Kaufmann, pages 1300 - 1309.

Fukushige, Y. (2004). Representing probabilistic knowledge in the semantic web. In *position paper for the W3C Workshop on Semantic Web for Life Sciences,* Cambridge, MA, USA, 2004.

Fukushige, Y. (2005). Representing probabilistic relations in rdf. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 4th International Semantic Web Conference (ISWC),* Galway, Ireland, November 2005.

Gao, M. and Liu, C. (2005). Extending owl by fuzzy description logic. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI05),* Hong Kong, China, November 2005, pages 562–567.

Glymour, C. (2001). The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology . The MIT Press.

Gruber (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, volume 5(2), pages 199-220.

Guo, H. and Hsu, W. (2002). A survey of algorithms for real-time bayesian network inference. In *Proceedings of the joint workshop on Real-Time Decision Support and Diagnosis Systems in AAAI/KDD/UAI 2002,* Edmonton, Alberta, Canada.

Haase, P. and Stojanovic, L. (2005). Consistent evolution of owl ontologies. In *Proceedings of the Second European Semantic Web Conference,* Heraklion, Greece, volume 3532, pages 182 -197.

Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: the state of the art. *Knowledge Engineering Review*, volume 18(1), pages 1-31.

Laamari, N. and Ben Yaghlane, B. (2007). Uncertainty in semantic ontology mapping: An evidential approach. In *Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty(ECSQARU'2007)* Springer Lecture Notes in Computer Science 4724, Hammamet, Tunisia, pages 418-429.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their applications to expert systems. In *Proceedings of the Royal Statistical Society,* pages 154-227.

McGuinness, D. (1996). Explaining Reasoning In Description Logics. PhD thesis, Rutgers University, New Brunswick, NJ.

McGuinness, D. (1999). Ontologies for Electronic Commerce. In *Proceedings of the AAAI'99 Artificial Intelligence for Electronic Commerce Workshop.*

Mitra, P., Noy, N. F., and Jaiswal, A. (2005). OMEN: A Probabilistic Ontology Mapping Tool. In *Proceedings of ISWC'05,* Galway, Ireland, pages 537-547.

Nagy, M., Vargas-Vera, M., and Motta, E. (2006). DSSim-ontology mapping with uncertainty. In *Proceedings of the International Workshop on Ontology Matching (OM-2006)*, Athens, Georgia, USA, pages 115–123.

Nardi, D. and Brachman, R. (2003). An introduction to description logics. In *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, pages 1-40.

Noy, N. and Klein, M. (2004). Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*,volume 6(4), pages 428-440.

Pan, R., Zhongli, D., Yu, Y., and Peng, Y. (2005). A bayesian network approach to ontology mapping. In *Proceedings of the Fourth International Semantic Web Conference, Ireland*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Pub. San Mateo, Ca, USA.

Pool, M., Fung, F., Cannon, S., and Aikin, J. (2005). Is it worth a hoot? Qualms about OWL for uncertainty reasoning. In *Proceedings of the workshop of Uncertainty Reasoning for the Semantic Web at the 4th International Semantic Web Conference (ISWC05),* Galway, Ireland, pages 1-11.

Russel, S. and Norvig, F. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Shafer, G. (1976). A Mathematical Theory of Evidence. *Princeton University Press*.

Smets, P. (1993). Belief functions: the disjunctive rule of combination and the generalized bayesian theorem. *International Journal of Approximate Reasoning,9*, pages 1–35.

Smets, P. (1997). Imperfect information: Imprecision - Uncertainty. In Motro, A. and Smets, P., editors, *Uncertainty Management in Information Systems. From Needs to Solutions*. Kluwer Academic Publishers, pages 225–245.

Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J., and Horrocks, I. (2005). Fuzzy OWL: Uncertainty and the Semantic Web. In *Proceedings of the International workshop on OWL: Experience and Directions (OWL-ED05),* Galway, Ireland, November 2005.

Trabelsi, W. (2007). Implementing and comparing exact inference algorithms in evidential networks. Master's thesis, Institut Supérieur de Gestion de Tunis Tunisia.

Uschold, M. and Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Record*, volume 33(4), pages 58-64.

Van Rijsbergen, C. (1979). Information Retrieval, 2nd edition. Dept. of Computer Science, University of Glasgow.

Vomlel, J. (1999). Methods of Probabilistic Knowledge Integration. PhD thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University.

Wache, H., Vögele, T., Visser, V., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S. (2001). Ontology-Based Integration of Information - A Survey of Existing Approaches . In *The Workshop on Ontologies and Information Sharing at the international joint conference on AI (IJCAI),*pages 108-117.

Williamson, J. (2005). Bayesian nets and causality: Philosophical and computational foundations. Oxford University Press.

Xu, H. (1995). Uncertainty reasoning and decision analysis using belief functions in the valuation-based systems. PhD thesis, Université Libre de Bruxelles.

Yang, Y. and Calmet, J. (2005). OntoBayes: An Ontology-Driven Uncertainty Model. In *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation, International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA/IAWTIC)*, Washington, DC, USA. IEEE Computer Society, pages 457–463.

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, volume 1, pages 3-28.