TAMPERE UNIVERSITY OF TECHNOLOGY

RIKU EISCHER

REASONING UNDER FUZZY VAGUENESS AND PROBABILISTIC UNCERTAINTY IN THE SEMANTIC WEB

Master of Science Thesis

Tarkastaja: Ossi Nykänen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkotekniikan
tiedekuntaneuvoston
kokouksessa 9.12.2015

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
RIKU EISCHER: Päättely sumean epämääräisyyden ja probabilistisen epävarmuuden alla semanttisessa webissä
Diplomityö, 43 sivua, 0 liitesivua
Marraskuu 2015
Pääaine: Hypermedia
Tarkastajat: Ossi Nykänen
Avainsanat: epävarmuus, sumea, probabilistinen, semanttinen web

Datan yhdistäminen useasta lähteestä, jotka eivat ole täysin luotettuja tuo haasteita datan automaattiseen prosessointiin. Tiedon esittäminen luonnollisella kielellä hankaloittaa entisestään laskentaa. Semanttisessa webissä monet henkilökohtaiset agentit joutuvat selviämään monenlaisesta tiedon epävarmuudesta. Kirjallisuudessa on kaksi päälähestymistapaa tiedon epävarmuuden mallintamiseen - sumeat ja probabilistiset. Nämä lähestymistavat mallintavat semanttisesti erityyppisiä epävarmuuden lajeja. Tämä diplomityö keskittyy lähestymistapoihin, jotka yhdistävät sekä sumeaa että probabilistista päättelyä yksissä puitteissa, jotta automaattiset agentit voisivat selviytyä molemmista epävarmuuden lajeista.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Infromation Technology
RIKU EISCHER : Reasoning under fuzzy vagueness and probabilistic uncertainty in the Semantic Web
Master of Science Thesis, 43 pages, 0 Appendix pages
November 2015
Major: Hypermedia
Examiner: Ossi Nykänen
Keywords: Uncertainty, Semantic Web, Fuzzy, Probabilistic

Combining data from many different sources or from sources that are not entirely trusted brings challenges to the automated processing of such data. Knowledge presented in natural language is another challenge for computing. In the semantic web, many applications such as personal agents need to be able to manage multiple kinds of uncertainty. There are two main approaches to modeling uncertainty in the literature - fuzzy and probabilistic. These approaches model semantically different types of uncertainty. This paper focuses on approaches that combine both fuzzy and probabilistic reasoning in one framework to provide automated agents the capability to deal with both types of uncertainty.

PREFACE

This thesis was done at Tampere University of Technology (TUT), in the department of Mathematics/Hypermedia in 2013-2015. It was a personal project and as such was not funded.

I would like to thank my supervisor Ossi Nykänen for his advice and patience during the process. Our often delightfully sidetracked conversations served to clarify the direction of the work and gave me inspiration. I would also like to thank my family and friends for their support and for proofreading the thesis.

Tampere, November 24^{th}, 2015

CONTENTS

TERMS AND DEFINITIONS

ABox	Assertional Box
DL	Description Logic
DLP	Description Logic Program
GCI	General concept inclusion
GTU	Generalized theory of uncertainty
GCL	Generalized concept language
KB	Knowledge base
NL	Natural language
MEBN	Multi-entity Bayesian network
MTheory	MEBN theory
MFrag	MEBN fragment
OWL	Web Ontology Language
RBox	Role Box
RDF	Resource Description Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language
TBox	Terminological Box

1. INTRODUCTION

The World Wide Web has had an enormous impact on the way people communicate and how businesses operate. It is the driving force behind the transformation into information society. The Web, and especially the recent explosion of handheld devices capable of browsing the Web, has changed the way we perceive computing. Prior to the widespread use of the Web, computing was mainly numerical processing, database systems, text processing etc. Now computing is becoming a gateway to information highways.

The Semantic Web is a natural evolution of the Web. The vision is that most if not all of the functions Web users do manually could and should be automated. If one wants to find information on a specific topic, the usual process involves typing in keywords into a keyword-based search engine and manually sorting through the pages retrieved. People have become so accustomed to keyword searches and the engines are very good at finding relevant documents that this is a very effective way of finding information, to the point that it might be faster to search the web than walk across the room to read the same information from a book.

But imagine if the user could simply ask an automated personal agent the query in a natural language. The agent would then find and select relevant pieces of information, intelligently combine them and perform necessary reasoning to answer the question, and answer in a natural language. This need not be limited to queries. An agent could, for example, book a hotel room by querying the user for preferences, then negotiate with multiple service provider agents for the best deal, and make a reservation after confirming the result with the user. All in a matter of seconds. This is the part of the Semantic Web vision most visible to consumers.

But that is not all, it is in fact businesses that stand as much to gain from the Semantic Web. Building supply chains is a demanding process which requires a lot of cooperation and integration. It is therefore expensive. In a world of Semantic Web, businesses could negotiate trade deals, partnership contracts, deliveries etc. automatically or semi-automatically without any prior contact. This drastically reduces overhead of large supply chains and enables them to react to changes rapidly. If one supplier fails to deliver for whatever reason, an automated agent could instantly negotiate a replacement supply.

The nature of the Web is open - anyone can access and provide most information.

This raises questions about the trustworthiness of information. As an example, Wikipedia the free encyclopedia, due to its openness, has regular "edit wars" over controversial topics. Typically politically or religiously motivated editors edit facts to better suit their viewpoint and counterarguers roll back the edits. Uncertainty regarding information need not be intentional either. There are many situations where terminology in one field is confusing to someone from another field.

Semantic agents would need to be able to resolve or manage any uncertainty about the data they encounter in order to be fully automated. Current Semantic Web standard technologies offer little for managing uncertainty. As a result, several groups have developed different approaches to dealing with uncertainty in the Semantic Web. The most prominent ones are probabilistic, typically Bayesian network based, and fuzzy logic and set theory based approaches. They each have developed tools and reasoners for describing uncertainty in a specific way. Other approaches such as those based on Dempster-Shafer belief theory or Rough Sets exist, but are outside the scope of this thesis.

In a real world situation, an agent might encounter several sources of semantically different uncertainty. A query in a natural language could be vague. If relevant information is expressed in natural language, as most information on the Web currently is, it could also be vague. If, for example, the natural language system is a speech recognition interface, all kinds of noise or difficult accent could result in uncertainty. The data retrieved might be incomplete or false.

The purpose of this work is to look at methods found in the literature for dealing with multiple kinds of uncertainty. More specifically, to look at two of the most common methodologies for representing uncertainty - fuzzy logic and probabilistic methods - and present ways they have been combined in the literature for dealing with uncertainty.

Chapter 2 discusses the semantic web, how semantic technologies could benefit users and businesses and some relevant languages and technologies. Chapter 3 discusses different types of uncertainty. In chapter 4, some common approaches to dealing with specific kinds of uncertainty are presented. Chapter 5 discusses methods that combine multiple different kinds of uncertainty in an application, and presents a hotel booking agent example.

2. THE SEMANTIC WEB

The original World Wide Web was visioned as a global repository of documents, with easy access from anywhere. The documents were intended to be written by humans, for humans. In a world where new information is produced at an ever increasing rate, automated processing of data becomes necessary. The Semantic Web is visioned as an extension to the World Wide Web that enables machines to not only retrieve and process the documents, but to capture the semantics of the information contained in the documents. [1] This allows machines to perform much more advanced processing tasks like logical reasoning and answering complex queries like "What is the distance between the capitals of Finland and Sweden?". Simple keyword based search engines might return pages related to Helsinki and Stockholm, but do not understand the actual question. A semantic query system could, in principle, infer the capital cities to mean Helsinki and Stockholm, take their related geolocation data, calculate the distance and answer in kilometers. [2]

Most information found on the Web is in a form that is different to non-web documents like company reports [2]. The text on the web is typically shorter, possibly comprising of single words or short phrases. Moreover, the positioning of the text is usually important, so an automatic parser would need to take HTML layout into consideration to fully understand the semantics. Because of this, most traditional natural language (NL) methods are poorly suited for the Web. The solution offered by the Semantic Web is to explicitly store the semantics in machine processable form. [2; 3]

Another way of looking at the Semantic Web is as a database. With linked data and the idea of sharing data and it's semantics, one might view the Semantic Web as a large distributed database run by many independent parties. Software agents can query this database for information they need and interact with other agents, thus contributing data to the system themselves. This has always been one of the primary goals of the Semantic Web vision, enabling the wealth of information stored in the web to be machine processable [1]. There are challenges to this task similar to those faced by researches of databases. When and where should reasoning be handled, for example? When an agent needs information, it has to perform queries, some of which might be expensive. In order for the system to scale properly for the web, the implementation needs to be efficient. [2; 3]

The Semantic Web vision presented in [1] describes a web where software agents perform most tasks humans need to do manually in the classical web. They do tasks such as searching for prices, reviews and availability of products, or supplying a keyword based search engine with relevant terms and sorting through the given documents to find a piece of information. All such tasks could be automated if the semantics of information was available for machine processing. Software agents can be seen as the most important application of semantic web from the users perspective. [1; 2; 3]

Still, most present day semantic technologies are driven by business to business data exchange needs [3]. Traditionally, Electronic Data Interchange (EDI) approaches are used to exchange information. They need experts to program and maintain each individual connection and are thus costly. The Semantic Web promises fast, automated exchange negotiation of common terminology and thus lowering the overhead and manual labor associated with business to business commerce. Negotiations, partnership contracts etc. can be carried out automatically or semiautomatically as the need arises. [3]

2.1 Ontology

To accomplish the task of representing semantics of information in machine processable form, the Semantic Web relies on ontologies. The term originates from philosophy, used in metaphysics concerned with identifying things that exist and describing them. The ontologies in the Semantic Web are documents that describe a domain. Typically they are finite lists of terms that describe concepts in the domain and the relationships between the concepts.

Ontologies can serve as mere vocabularies describing concepts and their relationships in a domain. They may be used to map concepts from different sources to give them a formal semantics. For example, a bookstore might get data from one publisher in a form where the author of a book is "author" and "creator" from another publisher. An ontology could describe the relationship of these concepts as equivalent.

Ontologies can also contain information about individuals - data in general terms. For example 'person' might be a concept in an ontology, and 'father(A,B)' might be a property relating individual persons A and B as father and child. Concepts in ontologies typically form concept hierarchies, e.g. 'penguin' might be a subclass of 'bird', which in turn could be subclass of 'vertebrate'. [2] This subclass relationship borrows terminology from object oriented programming. In logics terminology it is a subsumption or a subset relationship [4].

Other forms of information in ontologies may include properties (father(A,B), value restrictions (only a person may be a father), disjointness statements (father

and mother are disjoint) or logical relationships between concepts (a father must have at least one child). [3]

2.2 Layered stack of technologies

The development of the Semantic Web proceeds in steps, each building a layer on top of the previous one. The aim is to establish a layered stack of technologies akin to an OSI type model. The rationale behind this approach is that it is easier to form de facto standards, such as the hypertext markup language HTML is for the classical web, in small steps. Overview of the stack is presented in figure 2.1. Uniform resource identifier (URI) layer is the only part of the stack that is universally used in the classical web in the form of uniform resource locators URLs. Unicode and XML (XHTML, for example) are supported nearly universally in the web, but are not the only technologies. The Semantic Web would use these in a similar way to the current web, URIs to identify resources, Unicode as character encoding and XML to give structure to markup. [2; 3; 4]

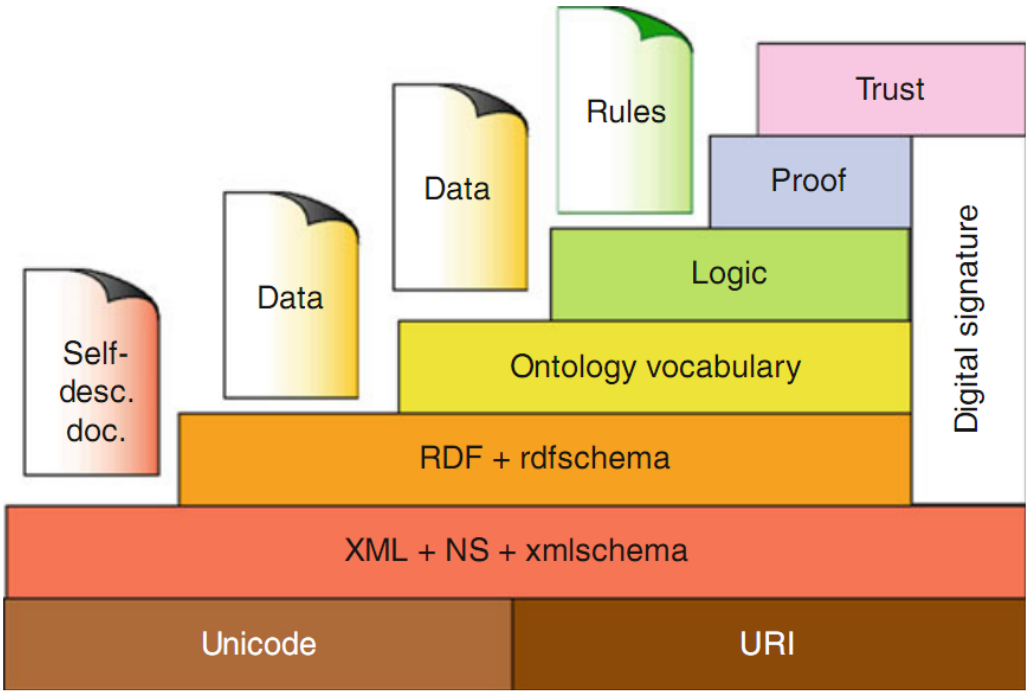

Figure 2.1: Semantic Web layers [2].

2.2.1 RDF

On the RDF layer the semantic web parts from the current web. Resource description framework RDF is a basic data model with XML-based syntax [3]. RDF

consists of resources, properties and statements. Resources are objects the statement is describing, and they each have a unique URI. Properties are a special kind of resource that describes relations between resources. As properties are also resources, they too have URIs. Statements store the information on properties of resources. They consist of a resource, property and a value, which can be either a resource or a literal (e.g. integer, string). [3] Here is an example of a simple RDF document with just one statement:

```
<?xmlversion="1.0"encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:example="http://www.example.org/ex-rdf-ns">

    <rdf:Description rdf:about="http://tut.fi">
        <example:owned-by rdf:resource="#TUT" />
    </rdf:Description>

<rdf:RDF>
```

In the example, URL is used to point to the domain tut.fi, it has the property owned-by and the object is a URI #TUT. One interesting property of RDF is reification, that is, it allows statements about statements [3]. This allows for statements such as "the Finnish Communications Regulatory Authority FICORA authorizes TUT to own tut.fi domain".

2.2.2 RDFS

RDF does not make any assumptions about any application domain nor does it define any semantics. For this, there is RDF schema (RDFS) [3]. To define semantics, RDFS separates classes, individuals and properties. A class is a collection of elements. Individual objects which belong to a class are instances of the class. RDFS enables construction of class and property hierarchies by defining rdfs:subClassOf and rdfs:subPropertyOf properties. It also introduces rdfs:domain and rdfs:range to restrict the subject and object of a property to specific resources.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <rdfs:Class rdf:ID="organization"/>
```

```
    <rdfs:Class rdf:ID="university">
        <rdfs:subClassOf rdf:resource="#organization"/>
    </rdfs:Class>

    <rdf:Class rdf:ID="student"/>

    <rdf:Property rdf:ID="studiesAt">
        <rdfs:domain rdf:resource="#student"/>
        <rdfs:range rdf:resource="#university"/>
    </rdfs:Property>

<rdf:RDF>
```

2.2.3 SPARQL

RDF triplestores are an effective way to store data. A database needs a query language and as RDF is at a higher level of abstraction than XML, using XML based query methods like XPath leads to problems. SPARQL is the query language for RDF. It uses a structure similar to the Structured Query Language SQL: SELECT specifies the projection, what data to retrieve; optional FROM specifies the source; WHERE imposes constraints on the query. [3] A simple example of a SPARQL query could be:

```
SELECT ?x ?y
WHERE
{
    ?x studiesAt ?y
}
```

2.2.4 OWL

The expressivity of RDF and RDFS is still fairly limited. RDF is roughly limited to binary ground predicates and RDFS is limited to class and property hierarchies and domain and range constraints. This led to research into more expressive ontology languages. The most important feature required by the semantic web but not provided by RDFS is support for reasoning. The Web Ontology Language OWL was designed to allow for reasoning tasks such as consistency checking, equivalence checking and classification. [3]

Reasoning brings new challenges to the design of ontology languages, however. Certain RDFS features such as rdfs:Class, the class of all classes, and rdf:Property,

the class of all properties allow very expressive statements that would lead to uncontrollable complexity [3]. Reification is another feature that poses problems for efficient implementation [2]. For this reason, the decidable dialects of OWL are not compatible with all RDF and RDFS statements like they should in a well behaved layer model. This breaks the layering - an agent using a decidable variant of OWL cannot use all lower layer information. This seems to be a fundamental problem with trying to layer decidable logic on top of RDF in OSI model style [2].

The first version of OWL has three sublanguages, OWL Full, OWL DL and OWL Lite. OWL Full does not pose any restrictions on the use of OWL primitives, including changing the meaning of OWL or RDF language primitives. This means OWL Full is fully compatible with RDF and RDFS, but this also means it is undecidable - no complete or efficient reasoning is possible. OWL DL was designed as a sublanguage of OWL Full that is as expressive as possible but decidable. It imposes restrictions on OWL and RDF constructors, essentially disallowing application of language constructors on each other. OWL Lite is an even more restricted subset of OWL Full and DL. The reasoning behind OWL Lite is that many powerful features in OWL are actually unnecessary in practice [3]. It should be noted that every OWL Lite ontology is a legal OWL DL ontology, and every OWL DL ontology is a legal OWL Full ontology. [2; 3; 4] The following OWL DL example highlights some features added by OWL compared to RDFS. Disjointness of assistant and professor, inverse property (teaches vs isTaughtBy) and property restriction (a course must have at least one isTaughtBy property):

```
<?xmlversion="1.0"encoding="UTF-8"?>
<rdf:RDF
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    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#">

    <owl:Ontology rdf:about="">

        <owl:Class rdf:ID="course"/>

        <owl:Class rdf:ID="staffMember"/>

        <owl:Class rdf:ID="professor">
            <rdfs:subClassOf rdf:resource="#person"/>
        </owl:Class>
```

```
    <owl:Class rdf:ID="assistant">
        <rdfs:subClassOf rdf:resource="#person"/>
        <owl:disjointWith rdf:resource="#professor"/>
    </owl:Class>

    <owl:ObjectProperty rdf:ID="isTaughtBy">
        <rdfs:range rdf:resource="#courser"/>
        <rdfs:domain rdf:resource="#staffMember"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:ID="teaches">
        <rdfs:range rdf:resource="#staffMember"/>
        <rdfs:domain rdf:resource="#course"/>
        <owl:inverseOf rdf:resource="#isTaughtBy"/>
     </owl:ObjectProperty>

    <owl:Class rdf:about="#course">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#isTaughtBy"/>
                <owl:minCardinality
                    rdf:datatype="&xsd;nonNegativeInteger">1
                </owl:minCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

</owl:Ontology>

<rdf:RDF>
```

OWL 2 is an update to OWL, which includes new features like increased expressive power for properties, extended support for datatypes, meta modelling capabilities, extended annotation capabilities and database style keys [4]. Similar to the previous version, OWL 2 also defines additional profiles as subsets of OWL 2 Full: OWL 2 EL, OWL 2 QL and OWL 2 RL. OWL 2 EL is aimed at applications that need to process very large numbers of classes and properties. It is a subset of OWL 2 for which efficient, highly scalable polynomial time algorithms exist. OWL 2 QL is for applications with large volumes of instance data where efficient query answering is the most pressing need. Conjunctive queries can be implemented effi-

$\mathcal{AL}$ Syntax	FOL translation	explanation
A	$A(x)$	atomic concept
$\top$	$\top(x)$	universal concept
$\bot$	$\bot(x)$	bottom concept
$\neg A$	$\neg A(x)$	atomic negation
$C \sqcap D$	$C(x) \wedge D(x)$	concept conjunction
$\forall R.C$	$\forall y.R(x,y) \rightarrow C(y)$	universal restriction
$\exists R.\top$	$\exists y.R(x,y)$	unqualified existential restriction

Table 2.1: Basic DL $\mathcal{AL}$ and its FOL translation.

ciently by translating them into standard relational query language like SQL. This results in OWL 2 QL having quite limited expressivity. Finally, OWL 2 RL has efficient reasoning algorithms: consistency checking, satisfiability, subsumption, instance checking and conjunctive query answering can be solved in polynomial time. OWL 2 RL reasoning systems can be implemented using logic programming, as mapping to Datalog exists. [4]

2.3 Description Logic

The Web Ontology Language (OWL) variants are based on description logics (DL) that provide a theoretical base for algorithms and computation for OWL. OWL languages can be mapped to their corresponding DL and computational complexity and algorithms found for the DLs can be applied to OWL. DL's are identified by a string of calligraphic letters that denote the extensions the DL adds to the basic DL $\mathcal{AL}$. To limit complexity, DLs use a small number of constructors to build complex concepts. The basic building blocks of DLs are concepts (unary predicates), roles (binary predicates) and individuals (constants). [4]

The syntax of the basic DL, $\mathcal{AL}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage), along with an informal translation into corresponding first order logic (FOL) statement is presented in table 2.1. In $\mathcal{AL}$, the so called TBox (terminological box) consists of a finite set of general concept inclusion (GCI) axioms. A GCI is of the form $C \sqsubseteq D$ intuitively means a subclass relation, e.g. every instance of C is also an instance of D. These can be used to order concepts into hierarchies. An ABox (assertional box) is a finite set of concept and role assertion axioms $a : C$ and $(a, b) : R$, where a and b are individuals. A knowledge base KB consists of a TBox T and an ABox A, $K = (T, A)$. [4]

Other DLs are formed by extending $\mathcal{AL}$ with additional constructs. Of special interest here is the DL $\mathcal{SROIQ}(D)$, which is the DL behind OWL 2 Full. $\mathcal{SROIQ}(D)$ adds concept negation and transitive roles ($\mathcal{S}$), complex role inclusion ($\mathcal{R}$), nominals ($\mathcal{O}$), inverse role ($\mathcal{I}$) and qualified number restriction ($\mathcal{Q}$) to the base language [4]. The (D) denotes concrete domains, which means OWL 2 can use

basic datatypes such as strings and integers, allowing concepts such as "Person $\sqcap$ $\exists$age.(xsd:integer:$\geq$ 20)", denoting persons equal to or over 20. For an example of a knowledge base, see section 5.2, axioms 5.3 - 5.6. As $\mathcal{SROIQ}(D)$ allows much more complex role axioms, a role box (RBox) is added to TBox and ABox to form a $\mathcal{SROIQ}(D)$ knowledge base. [4]

Reasoning in the DL family of languages typically reduces to KB satisfiability problem. For example, GCI or otherwise known as subsumption reduces to $KB \models C \sqsubseteq D$ iff $KB \cup \{a : C \sqcap \neg D\}$ is not satisfiable, where a is a new individual. For this, several variations of tableau algorithm have been developed for various DLs [6]. The tableau algorithms construct a forest of completion trees. The idea is to generate a finite interpretation by applying a set of transformation rules that branch out to create trees. The algorithm terminates when no rules apply. If the trees are free of contradictions, the KB is satisfiable. [4; 6]

2.4 Description Logic Programs

Description logic is a subset of first-order logic for which efficient proof systems exist. Another subset is Horn logic, or so called rule systems [5]. They are orthogonal in the sense that neither is a subset of the other. In Horn logic it is possible, for example, to express statemets such as "A is the uncle of B if A is the brother of B's father C". Such statements are impossible in OWL. Horn logic, on the other hand, can't express negation/complement, disjointness or existential quantification. [3; 4]

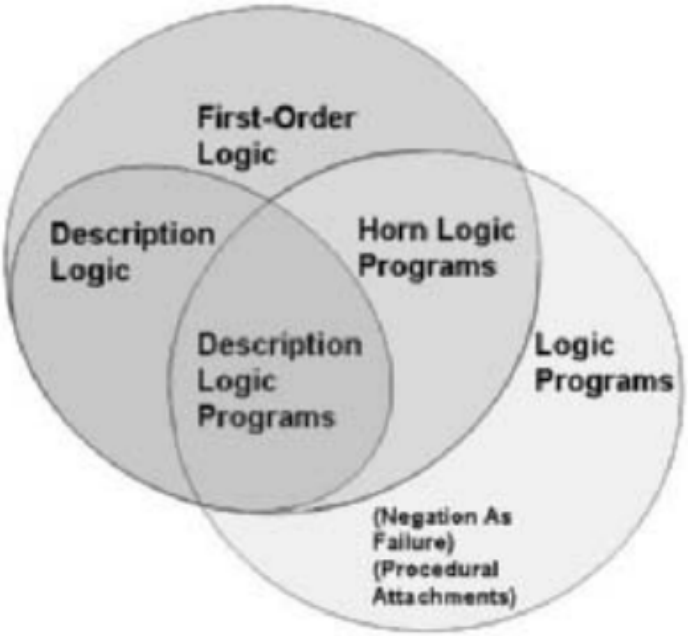

Figure 2.2: Relation of DLP to other logics [3].

Typically rule systems have statements of the form $B \leftarrow A_1, ..., A_n$, where A_i and B are atomic formulas. The previous uncle example could be:

$$uncle(A, B) \leftarrow brother(A, C), childOf(B, C) \tag{2.1}$$

There are two ways to interpret rules, deductive and reactive. In deductive interpretation, if $A_1, ..., A_n$ are known to be true, then B is true. Reactive interpretation, on the other hand, states if the conditions $A_1, ..., A_n$ are met, then do B. Both have important applications. [3]

Description Logic Programs are the intersection of Horn logic and description logic, see figure 2.2. In other words, DLP are the OWL-definable part of Horn logic, or vice versa. As a consequence, either OWL or rule system tools and reasoners can be used. Experience says existing ontologies rarely use OWL constructs that are outside the DLP subset [3].

3. UNCERTAINTY

The World Wide Web and by extension the semantic web are difficult environments for automated systems, due to the unregulated sharing of data. Essentially anyone can say anything about anything (AAA). In this context, automated systems need to be able to cope with data of various quality.

Uncertainty, in the context of Semantic Web data, is a broad concept used here and in the literature to describe information that is partially unknown, unknowable, untrustworthy or missing. The World Wide Web Consortium (W3C) held an incubator group on uncertainty reasoning in the Semantic Web in 2008. The aim of this group was to identify and describe cases where reasoning with uncertain information would yield better results than conventional methods. [7]

The W3C Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) identified different forms of uncertainty, how and why they appear in the data and some ways to model them. The group released an ontology describing these properties, available at [8]. The ontology differentiates how the uncertainty is derived, what kind of uncertainty it is and what is it's nature. The derivation is either objective, that is, a formal process causes the uncertainty, or subjective - a guess or judgement. The nature is either aleatory - uncertainty is a property of the world - or epistemic, where the uncertainty comes from the agent's limited knowledge. The types of uncertainty are ambiguity, empirical, randomness, vagueness, inconsistency and incompleteness, which are discussed in more detail in the following sections. [7]

It should be noted that the aforementioned grouping can be viewed as special cases of two sources of uncertainty: incomplete information in which the information is well defined but parts of it are missing, and vague information where it is not precisely defined. These correspond roughly to probabilistic models which deal with incomplete information and fuzzy models that deal with vague information. There is overlap and it is not clear which family of reasoning would be better in practice. Especially in the context of the Semantic Web a large source of uncertainty comes from the requirement of semantic web agents to be able to process natural language information.

3.1 Ambiguity

The first type of uncertainty mentioned in [7] is ambiguity. Ambiguity refers to a situation where there are two or more distinct interpretations for a concept, and it is not clear which is applicable. Natural language is full of situations where one word or phrase may have multiple meanings. It is usually clear, for a human, which meaning is the correct one in each situation. However, such cases present problems for automated processing of data.

In the context of the Semantic Web the problem mainly occurs in ontology alignment. Two distinct ontologies may have same labels for completely different concepts. A great example is given in [9] where the group aligned two large thesauri, the UN's Food and Agriculture Organization's AGROVOC and US's National Agricultural Library's NALT thesauri. One source of error in the alignment process was the mapping of "Game" into a single concept. In one thesaurus Game refers to a sports activity, and a wild animal in the other.

The problem of semantic ambiguity is solved in many cases by using multiple similarity measures, usually syntactic and semantic similarity [10]. Such systems will not only consider the syntactic similarity, but also the semantic graph similarity. That is, they also consider if the concepts have similar ancestors and children, etc. By examining the graph it becomes clear that these two concepts use the same label but are very different conceptually. Note that this example of ambiguity has objective derivation but epistemic nature - an agent would always arrive at this problem given it doesn't have enough knowledge to resolve the ambiguity.

3.2 Empirical

Empirical uncertainty refers to the situation where a sentence has a truth value but it is not known until more information is obtained. Consider the statement "The grass is wet when it rains", or wet grass is a logical consequence of rain. Without any knowledge of rain it is uncertain whether the grass is wet or not. If more evidence is obtained, namely rain is true, then the uncertainty vanishes (note that if rain is false we still do not know the state of the grass). This can be seen as a special case of incompleteness. Probabilistic methods have a long tradition for dealing with this kind of uncertainty.

3.3 Randomness

Uncertainty related to randomness occurs when a statistical law determines the truth value of a sentence. For example a sentence concerning a fair coin flip could state "the coin lands heads". The truth value concerning the class of coin toss would only realize when instantiated as an actual toss, which has a 0.5 probability to be

true in our case. This might be viewed as a special case of empirical uncertainty; if we knew the exact state of the coin tossing system, it would cease to be random. If the exact state of both the momentum and location of a particle cannot be known under Heisenberg uncertainty principle, the best we can do is state probabilities. That serves to highlight the fact that our information about the real world is and likely always will be incomplete.

3.4 Vagueness

Vague information is lacking in precise definition, typically resulting from borderline cases. Species in biology are an example of vaguely defined concepts, where it is often impossible to determine when a separated group has evolved to a new species. Traditional computing or analytical methods inherently cannot model or reason with vague information, but it is commonplace and sometimes even beneficial in natural language. In order to model vague information a theoretical framework to model vague concepts in precise terms is needed. Fuzzy logic is one such framework and can be adapted for use in Semantic Web applications. Note that probabilistic methods are not easily applicable in the case of vague information.

As an example, consider the natural language terms tall and short. We can say a 1.9 meter person is tall and 1.5 meter is short, but how about 1.7 meters in height? We could say 1.7 meter is average, but then there would be the problem of 1.6 meters, is she short or average? Framing this problem in probabilistic terms makes little semantic sense, a 1.6 meter person is not short with probability 0.5 and average with probability 0.5, but rather both to some degree. There are ways around vagueness, for example binning below below 1.6 as short, 1.6-1.8 as average and above 1.8 as tall, but such techniques might not always be possible or beneficial. Fuzzy methods can directly capture vague expressions prevalent in natural language and reason with them.

3.5 Inconsistency

Maintaining consistency can be a problem in large knowledge bases or especially when combining two or more knowledge bases. Inconsistent knowledge can't be reasoned with and any inconsistencies must be resolved before the knowledge becomes useful. Checking for consistency is a basic operation in all reasoning systems. A knowledge base (KB) is consistent iff there are no contradictions. A contradiction refers to the situation where both proposition P and its negation are provable from the axioms of KB. Another definition for consistency is that KB is consistent iff there exists an interpretation I where all axioms of KB are true.

An example of an inconsistent KB might be one that states 1) birds can fly 2)

penguins are birds 3) penguins can't fly. A penguin can both be proven to fly and not fly using these axioms. It is possible to model this kind of contradicting information in some types of description logics using local consistency where a more specific piece of knowledge takes precedence. In this example, penguin is a subclass of bird and thus more specific. A statement concerning penguins is more specific than the one concerning all birds, and the KB would effectively state birds can fly, except if it's a penguin, then it can't fly. [11]

3.6 Incompleteness

A situation where some information is missing. Most systems cannot function without vital information, but it is a common situation in the real world. It is a hot topic in artificial intelligence [2; 4] where systems might not have the luxury of simply waiting for information and a decision needs to be made in real time. Both empirical uncertainty and randomness can be seen as special cases of incomplete knowledge. Probabilistic reasoning at its core deals with incomplete information.

4. UNCERTAINTY MODELS

The main approaches in the literature [7; 11] for handling uncertain data are fuzzy logic based and probabilisticl models. Although all models examined here use a value in the range $[0, 1]$ to model the uncertainty related to a statement, care must be taken to distinguish how they should be interpreted. There have been some misconceptions in the literature, and a clarification is provided by [12]. A fuzzy logic model might interpret truth value of 0.5 associated with rain to mean moderate rainfall. A probabilistic model might use a probability value of 0.5 associated with rain to denote a fifty per cent probability of rain. Probabilistic differs from possibilistic in that probability is the sum of all probabilities in worlds that satisfy an event, whereas possibility is the maximum probability among worlds that satisfy an event [11].

4.1 Fuzzy

Fuzzy logic dealing with fuzzy sets is a truly many-valued logic that can be used to model partial truth. It is a precise logic reasoning on imprecise knowledge. The main advantage of fuzzy models which applies to many fields is that they are intuitive and easy to interpret. This is mainly achieved by using linguistic variables and if-then rulebases. For example the variable temperature might have values cold, warm and hot. The rulebase becomes intuitive to define, e.g. IF temperature IS hot THEN turn on cooling. This is the most widely used aspect of fuzzy logic.[4; 13]

According to L.A. Zadeh [13] fuzzy logic has many more powerful features. It is a more general theory than bivalent (two-valued) logic and any bivalent logic can be generalized by adding fuzzy methods to it. This includes most known, possibly all probabilistic methods [13]. Another powerful feature is that Fuzzy methods are applicable to natural language (NL) information. Natural language describes perceptions (e.g. "it is quite warm") where the exact temperature is unknown, but an agent has described his perception of temperature. Fuzzy logic offers methods that can directly be used for reasoning with such vague information, given the definitions of quite and warm are calibrated for the agent doing the reasoning. They do not need to be exactly aligned with the definitions of the agent describing the perception. This is the powerful feature of NL of tolerating imprecision and slightly misaligned definitions, but still conveying essential information.

	t-norm $a \otimes b$	s-norm $a \oplus b$	negation $\ominus a$
Lukasiewicz	$max(0, a+b-1)$	$min(1, a+b)$	$1-a$
Gödel	$min(a, b)$	$max(a, b)$	1 if $a = 0$, 0 otherwise.
product	$a \cdot b$	$a+b-ab$	1 if $a = 0$, 0 otherwise.

Table 4.1: Common fuzzy set and logic operations.

4.1.1 Fuzzy Logic and Set Theory Basics

Fuzzy sets are a generalization of classical set theory to many-valued sets. An element is not either part of a set or not, but rather is a member to a degree. Usually this is denoted by a membership function $\mu_A : X \to [0, 1]$. The function maps X's membership to fuzzy set A to the interval $[0, 1]$. Typical membership functions are triangle, trapezoid, left-shoulder and right-shoulder, depicted in figure 4.1. Standard set operations can also be generalized to many-valued sets. There are, however, several variations of set operations, each of which fulfill the defining axioms for the operation but give slightly different results. Conjunction is generalized as triangular norm or t-norm denoted $\otimes$, disjunction is t-conorm or s-norm $\oplus$ and negation is fuzzy negation $\ominus$. Common operations are presented in table 4.1. [4]

The types of membership functions, their number and parameters as well as types of fuzzy operations all need to be selected for each application. Certain combinations work better in certain situations and it is not trivial to find the best suited ones for each application. This is why there are several variations of each. [14]

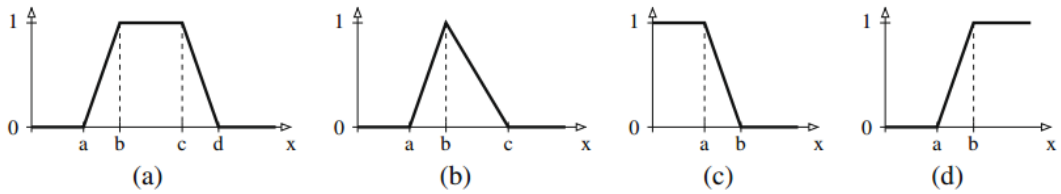

Figure 4.1: Fuzzy membership functions: (a) Trapezoidal function $trz(a, b, c, d)$, Triangular function $tri(a, b, c)$, Left-shoudler function $ls(a, b)$, Right-shoulder function $rs(a, b)$ [4].

Fuzzy logic maps truth values to $[0, 1]$ instead of $\{0, 1\}$. The same conjunction, disjunction and negation operations can be used as with fuzzy sets. On top of that, fuzzy implication is also defined. Lukasiewicz, Gödel and product logic are again major players, and an implication is defined for each of them: $a \Rightarrow_l b = min(1 - a + b, 1)$ for Lukasiewicz, $a \Rightarrow_g b = 1$ if $a \leq b$, b otherwise for Gödel and $a \Rightarrow_p b = min(1, b/a)$ for product logic. [4] For more information the reader should consult a fuzzy logic textbook, for example [14] or [4].

4.1.2 Reasoning with Fuzzy DLs

Like the most expressive OWL variants, some fuzzy DLs have a problem with decidability. More specificially when description logic is extended with fuzzy logic with infinitely many truth values (infinite model, e.g. [0,1]), it can be shown to be undecidable if general concept inclusions (GCI) are also allowed. GCI's are DL term for OWL subclass relation. [15] This has led to using finite fuzzy models, which are decidable with GCI's. That is, models with truth space of finite granularity, e.g. {0, 0.25, 0.5, 0.75, 1.0}. An additional benefit for this is that these finite fuzzy models are reducible to crisp KBs. This implies that only a fuzzy middleware translating a fuzzy KB into the corresponding crisp KB is needed and standard crisp reasoners can be used on the resulting KB.

Several fuzzy reasoners have been implemented for various fuzzy DLs. The most notable are FuzzyDL, FiRE and DeLorean. [4] FuzzyDL is a fuzzy $\mathcal{SHIF}(D)$ reasoner written in java. It extends the DL $\mathcal{SHIF}$ with fuzzy set operations, offering Gödel, Lukasiewicz and Zadeh fuzzy set operations, and concrete data types such as integers, reals, strings and fuzzy membership functions. Truth values are restricted to a finite model. Reasoning combines a fuzzy version of the standard tableaux algorithm with multi integer linear programming (MILP) optimization, allowing certain linear inequation constraints. The reasoner is operated via a simple query interface using FuzzyDL's own syntax, e.g. greatest lower bound query: (max-sat? C [a]). [16]

DeLorean is a fuzzy $\mathcal{SROIQ}(D)$ reasoner that translates a finite truth valued fuzzy KB into a crisp KB. It then uses standard crisp reasoner to perform the reasoning. [17] FiRE, on the other hand, is a fuzzy-$\mathcal{SHIN}(D)$ reasoner that serializes a finite truth valued fuzzy ontology into crisp RDF. FiRE works by materializing all implicit knowledge and storing it explicitly in RDF, then supplying SPARQL queries to it. This means it does not perform complete f-$\mathcal{SHIN}(D)$ reasoning, but is complete for ground conjunctive queries, which the authors note is a common practice for proposed practical applications. [18]

4.1.3 Fuzzy OWL 2

All the reasoners mentioned in the previous section use their own syntax to represent fuzzy ontologies. This means in order to use existing knowledge it needs to be typed in manually using the reasoner syntax or using automated translation, which might not exist for a given ontology language. [19] proposes a standard way to model fuzziness in OWL 2 using annotation properties. This means standard ontology tools such as Protégé can be used. Manually annotating the fuzziness can be tedious and error prone, and for that a Protégé plug-in Fuzzy OWL 2 is freely available on the

Web [20]. It supports fuzzy datatypes, fuzzy modified concepts, weighted concepts, weighted sum concepts, fuzzy nominals, fuzzy modifiers, fuzzy modified roles, fuzzy axioms, fuzzy modified datatypes, as well as provides parsers that translate Fuzzy OWL 2 into languages supported by some fuzzy reasoners. [4] Here is an example in Fuzzy OWL 2, which says Alice is a person who is tall to a degree of at least 0.7:

```
<Declaration>
    <Class IRI="#Person"/>
</Declaration>
<Declaration>
    <Class IRI="#Tall"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Alice"/>
</Declaration>
<ClassAssertion>
    <Class IRI="#Person"/>
    <NamedIndividual IRI = "#Alice"/>
</ClassAssertion>
<ClassAssertion>
    <Annotation>
        <AnnotationProperty IRI="#fuzzyLabel"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">
            <fuzzyOwl2 fuzzyType="axiom">
                <Degree value="0.7" />
            </fuzzyOwl2>
        </Literal>
    </Annotation>
    <Class IRI="#Tall"/>
    <NamedIndividual IRI="#Alice"/>
</ClassAssertion>
```

Noteworthy here is that everything is done using OWL 2 syntax, without the need for even an upper ontology. The <fuzzyOwl2> statement is inside <Literal> and carries no semantic information for normal OWL 2 reasoners. A Fuzzy OWL 2 aware reasoner can, however, use this information.

4.2 Probabilisitc

Probability theory has a long tradition in dealing with incomplete information. Indeed, probability is taught at almost all levels of education and the reader should

be familiar with the terminology. Many methods concerning the semantic web use Bayesian probability theory [21; 22; 23], although some use different approaches [24], [25]. Bayesian networks are a probabilsitic directed acyclic graphical model (DAG), where nodes represent random variables and edges represent conditional dependencies. Each node is associated with a probability function that takes the node's parent variables and outputs the node's probability distribution. See for example [26] for complete explanation.

As it has been with fuzzy, representing uncertain information in an ontology has been an active research topic with probabilistic methods. Some older approaches extend older ontology langauges such as [24] for DARPA Agent Markup Language and Ontology Inference Layer (DAML+OIL), but these ontology languages are superseded by OWL and OWL 2. Slightly newer approaches [22; 23] extend OWL with additional classes that allow them to convert the ontology into Bayesian networks and do probabilistic reasoning. The most notable probabilistic approach is PR-OWL [21], as it is the most complete and expressive. PR-OWL is covered in more detail in the next section.

4.2.1 PR-OWL

Usual solution for representing uncertainty in languages that do not support it natively is using custom tags to annotate statements with probability information. This solution is simplistic and is unsuitable for all but the simplest real world applications [21; 27]. Shafer stressed that probability is more about structure than numbers [28].

PR-OWL is a probabilistic extension to classical deterministic logic of OWL, providing full first order probabilistic logic. It is a Bayesian approach, using multi-entity Bayesian networks (MEBN). Bayesian probability theory is a well established framework for reasoning under uncertainty. MEBN combines Bayesian probability theory with classical first order logic (FOL). PR-OWL is an upper ontology, and as such is not a direct extension of OWL language, but defined as an ontology that can be used to create probabilistic ontologies. An ontology using PR-OWL may have a probabilstic part using PR-OWL deginitions, forming an MEBN theory, and a normal OWL part - there is no need for every statement to have probabilstic definitions.[21]

In a MEBN, knowledge is expressed as MEBN fragments (MFrags), which are organized into MEBN theories (MTheory). The MTheory is a collection of MFrags that satisfy consistency and have a unique joint probability distribution. An MFrag contains hypotheses related to a concept the MFrag is modeling and forms a conditional probability distribution of its resident random variables (RV). It can be instantiated as many times as necessary. An example could be "Genetic Disease"

MFrag, containing RVs about risk factors. It could be instantiated for each patient, giving information about risks based on the probabilities defined in the MFrag. Different hypotheses, or RVs, in an MFrag can be context (assertion that must be satisfied for the definitions in the MFrag to apply), input (probabilities defined in another MFrag) or resident (defined in the MFrag in question). [21]

In MEBN inference, a query is posed to compute the degree of belief in a random variable (RV), given a set of evidence RVs. In our previous genetic disease example, the evidence might be known risk genes that have been determined on the patient. A situation specific Bayesian network (SSBN) is created to address each query, which is a Bayesian network that begins with the MFrag associated with queried RVs. It then recursively instantiates the necessary MFrags required to calculate any distributions or values another MFrag needs. [21] In order to add these features to OWL, PR-OWL adds several new concepts, shown in figure 4.2.

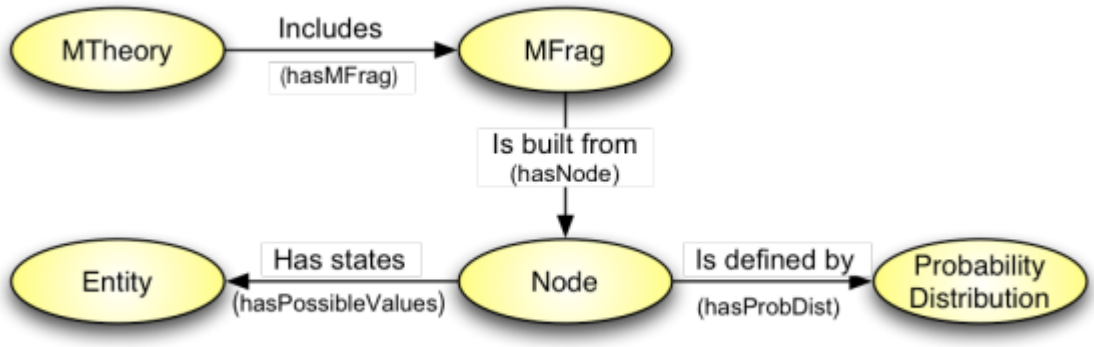

Figure 4.2: Overview of concepts added by PR-OWL [21].

A PR-OWL ontology must have at least one MTheory consisting of MFrags that form a valid MEBN theory. The link between MTheory and its MFrags in PR-OWL syntax is done via the object properties hasMFrag and its inverse, isMFragIn. MFrag instances are a collection of Nodes representing RVs, which are MEBN hypotheses that can be context, input or resident discussed earlier. Each Node is a random variable with a set of states that is mutually exclusive and exhaustive (add up to probability 1), depicted as hasPossibleValues. They also have a probability distribution, denoted by hasProbDist linking it to an instance of Probability Distribution class.[21]

PR-OWL can express a probability distribution on any interpretations of first-order logic. As a result of PR-OWLs expressiveness, the authors note that there are no guarantees for efficiency or decidability. This is a design choice in line with W3Cs OWL, where the main language is very expressive and several sublanguages are defined to limit the expressivity to achieve efficient algorithms and decidability.

The authors of PR-OWL propose a sublanguage PR-OWL Lite could be developed to limit the language to expressions that have known efficient algorithms. [21]

The authors of PR-OWL [21] admit that building coherent MTheory is a difficult, error prone manual process requiring deep knowledge of Bayesian logic and PR-OWL. Tools such as UnBBayes make building probabilstic ontologies easier [27]. UnBBayes tries to combine familiar features from standard industry modeling approaches such as UML into probabilistic ontology creation process.

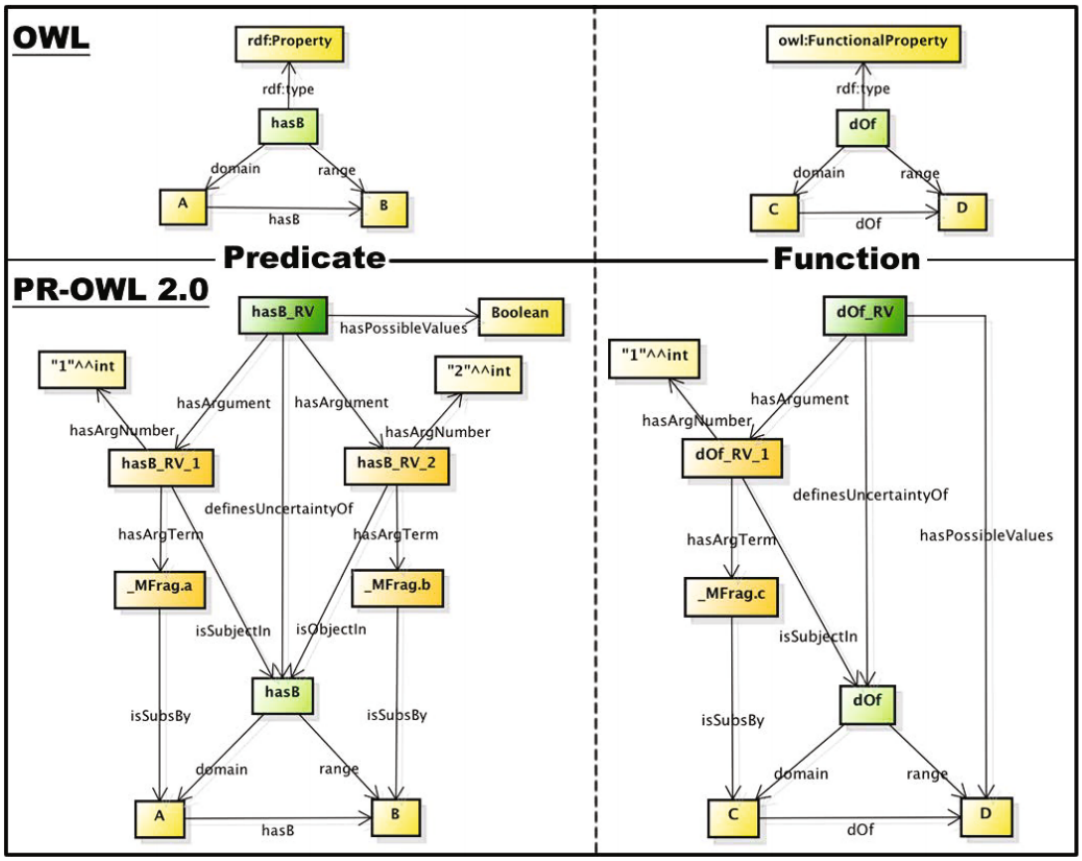

Figure 4.3: PR-OWL 2 - OWL mapping [29].

One major shortcoming identified in PR-OWL was the inability to map the base OWL properties into PR-OWL random variables. One might want to take knowledge from an existing OWL ontology and reason probabilistically with it using additional information from a PR-OWL ontology. There is no way to do this with PR-OWL, and thus PR-OWL 2 was developed to allow mappings of PR-OWL random variables and OWL properties. [29] Even the simplest of raw PR-OWL 2 documents get difficult to read without dedicated tools such as UnBBayes. Figure 4.3 shows the mapping of a single OWL property to PR-OWL 2. Here are some snippets from an ontology stating Alice has a child, Bob:

```
<ClassAssertion>
    <Class IRI="#Person"/>
    <NamedIndividual IRI="#Alice"/>
</ClassAssertion>
<Declaration>
    <ObjectProperty IRI="#hasChild"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Alice"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Bob"/>
</Declaration>

...

<ClassAssertion>
    <Class IRI="#Person"/>
<NamedIndividual IRI="#Alice"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Person"/>
    <NamedIndividual IRI="#Bob"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainMFrag"/>
    <NamedIndividual IRI="#Domain_MFrag.hasChild_MF"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainResidentNode"/
    <NamedIndividual IRI="#Domain_Res.hasChild"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#SimpleMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_hasChild"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#RandomVariable"/>
    <NamedIndividual IRI="#RV_hasChild"/>
```

```
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MappingArgument"/>
    <NamedIndividual IRI="#RV_hasChild_1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MappingArgument"/>
    <NamedIndividual IRI="#RV_hasChild_2"/>
</ClassAssertion>

...

<ObjectPropertyAssertion>
    <ObjectProperty IRI=
        "http://www.pr-owl.org/pr-owl2.owl#isMFragOf"/>
    <NamedIndividual IRI="#Domain_MFrag.tall_MF"/>
    <NamedIndividual IRI="#Tallness_MT"/>
</ObjectPropertyAssertion>

...

<DataPropertyAssertion>
    <DataProperty IRI=
        "http://www.pr-owl.org/pr-owl2.owl#hasDeclaration"/>
    <NamedIndividual IRI=
        "http://www.pr-owl.org/pr-owl2.owl#hasChild_Table"/>
    <Literal datatypeIRI="&xsd;string">
        [
        true = 0.5,
        false = 0.5
        ]
    </Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI=
        "http://www.pr-owl.org/pr-owl2.owl#definesUncertaintyOf"/>
    <NamedIndividual IRI="#RV_hasChild"/>
    <Literal datatypeIRI="&xsd;anyURI">
        http://www.example.org/example-ontology#is</Literal>
```

```
</DataPropertyAssertion>

...

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMFragOf"/
    <NamedIndividual IRI="#Domain_MFrag.hasChild_MF"/>
    <NamedIndividual IRI="#hasChild_MT"/>
</ObjectPropertyAssertion>
```

The entire listing is several pages longer. Noteworthy here are the complexity added by PR-OWL 2, requiring various mappings of OWL properties to MEBN RVs (RV_hasChild_1 and RV_hasChild_2 for example). Even the simplest possible example, containing no real information, is lengthy and complicated, and serves to highlight the vital importance of tools such as UnBBayes. Figure 4.4 shows the UnBBayes view of the example. On the left you can see the evidence that hasChild(Alice,Bob)=true. Otherwise the probability is defined as $true = 0.5, false = 0.5$, thus effectively stating that Bob has a child called Alice with probability 0.5. The purpose of this example was to take the simplest possible MEBN and demonstrate the PR-OWL 2 syntax. For a functional example, see [21; 29].

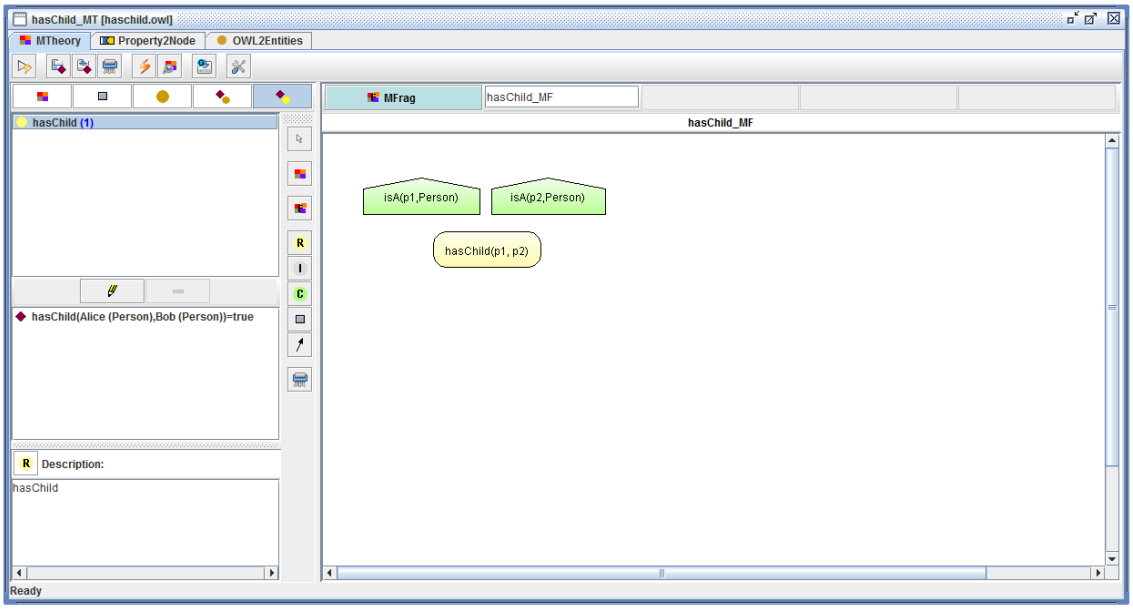

Figure 4.4: UnBBayes View of the example in section 4.2.1.

5. HYBRID APPROACHES

A major task in the Semantic Web is, given a query, access and retrieve relevant information from various different resources effectively. Once methods of modeling uncertainty mature and are used in real world applications, a common situation might be that some information is defined as probabilistic, e.g. in PR-OWL, and some is vague, e.g. Fuzzy OWL 2. How can agents deal with situations involving both uncertain and vague data?

Some methods are backwards compatible to their parent language OWL, so an agent not aware of probabilistic or fuzzy parts of the definition can still use the parts written in OWL [19; 21]. This way an agent utilizing probabilistic reasoning could still use fuzzy ontologies by discarding the fuzzy definitions unknown to it. This will lead to a loss of information on some level. It is also not clear how useful crisp vague definitions would be to a system that can not handle such vagueness. An example could be that a fuzzy ontology defines person A is tall to a degree of 0.7 and person B is tall to a degree of 0.2. A probabilistic reasoner that only reads the OWL parts and discards the fuzzy membership statements would conclude they both have the property tall, although person B is clearly quite short. This is also a problem in fuzzy control systems that need to convert the fuzziness into a crisp control signal [14].

5.1 Booking agent example

A classical example in the Semantic Web is that of a shopping agent [1; 2]. A shopping agent is a client side program that is given a query, and the task is to retrieve information about products that are relevant to the query and present the best results to the user. To illustrate some difficulties in combining information from various sources and using it to answer a vague query containing users preferences, a simple example of a hotel booking agent is presented. Let's say the user is interested in booking a high quality room with a balcony for no more than around 150€a night. For the sake of simplicity let's ignore aspects of availability and date of booking and assume the rooms the agent retrieves happen to be available on the date the user wants to make the booking.

There are two sources of semantically different uncertainty in this scenario. First, different hotels might use different ontologies to describe their available rooms. This

leads to a problem of ontology alignment, with inherent uncertainties. Addressing this process of alignment is outside the scope of this thesis, but let's assume the alignment produces probabilistic results about the similarity of concepts. One hotel might have in it's ontology a property hasBalcony associated with it's rooms. Another hotel might have hasFrenchBalcony. Are they equivalent concepts when the agent combines the information form the two hotel ontologies? The agent might conclude they are the same concept with a probability of 0.4 for example.

Second form of uncertainty comes from vagueness in the query. High quality room could for example mean the hotel has a rating of 4 or 5 stars and high reviews. These could be expressed as fuzzy membership functions. If the agent gathers user review data from a 3rd party site that grades hotels on a 1-10 scale, the whole high quality concept could be defined as $HighQuality(x) = HighStars(x) \sqcap HighReviews(x)$, where $HighStars(x)$ assigns membership of 0.7 for 4 and 1.0 for 5 stars, 0 otherwise, and $HighReviews(x)$ is a right-shoulder membership function starting from 0 at 7.5 and reaching full 1.0 at 8.5. Now, a hotel with 4 stars and average review score of 8.1 has $HighQuality$ membership degree (using min t-conorm) of $min(0.7, 0.6) = 0.6$.

Another fuzziness comes from the pricing, which was defined as no more than around 150€a night. This could be expressed as a left-shoulder membership function, which decreases the rooms desirability rapidly as the price goes above 150€: $ls(150, 175)$. The following sections discuss some methods found in the literature for dealing with these two semantically different types of uncertainty, and show how they can be used to address the hotel booking agent example.

5.2 Probabilistic Fuzzy Description Logic Programs

Probabilistic fuzzy DL programs as proposed by [25] take a stratified approach to combining probabilistic uncertainty and fuzzy vagueness in a uniform framework. Both fuzzy and probabilistic reasoning have separate well defined areas where they are applied in this query system. As probabilistic uncertainty and fuzzy vagueness are semantically very different, it is not sensible to use them interchangeably, but rather in their specific areas. The authors of [25] present an overview of a query system with a shopping agent example and give algorithms for processing the queries as well as proof of their polynomial time complexity under certain assumptions. [25]

The query system assumes the query agent and all relevant distributed resources each have their own ontologies, ontology languages and query languages. Major tasks are to 1) select relevant resources 2) reformulate queries from agent's query language to the target resources query language and 3) merge the results. While these problems have been studied in the literature, see [30] for resource selection and [31] for ontology alignment, this approach combines the use of probabilistic and fuzzy methods in a novel way. More specifically it uses probabilistic reasoning for resource

selection and ontology alignment, and fuzzy logic for vague query matching.[25]

At the core of reasoning, the query system uses normal fuzzy programs, a finite collection of normal fuzzy rules combined with description logic knowledge base. The normal fuzzy rules are similar to crisp normal rules like

$$NiceSportsCar(x) \leftarrow madeIn(x, y), Italy(y), DL[SportsCar](x) \tag{5.1}$$

with the exception that they have a lower bound for their truth value and use fuzzy conjunction rather than crisp ones. The $DL[SportsCar](x)$ is a query to description logic, asking if x can be proven to be a sports car from the knowledge base. A fuzzy version of the above rule could be:

$$NiceSportsCar(x) \leftarrow^{0.8}_{\otimes} madeIn(x, y) \otimes Italy(y) \otimes DL[SportsCar](x) \tag{5.2}$$

which informally means the head of the rule $NiceSportsCar(x)$ is at least the evaluated degree of the body. The degree of the body is the conjunction of it's components together with 0.8. A fuzzy dl-program$KB = (L, P)$ consists of fuzzy DL knowledge base L and a finite set of fuzzy dl-rules P. [25]

Probabilistic fuzzy dl-programs are a combination of stratified fuzzy dl-programs with Poole's independent choice logic [25, see 22]. Stratified fuzzy dl-programs are hierarchical construction of positive fuzzy dl-programs linked with default-negation. Positive fuzzy dl-programs are fuzzy dl-programs without any negation operations, which are always satisfiable and have a unique least model. The probabilistic fuzzy dl-programs define a probability distribution on a set of fuzzy interpretations. A probabilistic fuzzy dl-program is defined as $KB = (L, P, C, \nu)$, where (L, P) is the fuzzy dl-program, C is the set of random variables and ν is the probability distribution of C. [25]

Returning to the booking agent example, suppose an automatic ontology alignment between the user agent and hotel ontology produces the following alignments and associated probabilities: *hasBalcony* and *hasFrenchBalcony* are the same concept with probability 0.4, *hasPrice* and *pricePerNight* are the same with probability 0.95. This gives a total choice space C with two alternatives, $C_1 = \{BAL_{pos}, BAL_{neg}\}$, $C_2 = \{PRC_{pos}, PRC_{neg}\}$, along with probability distributions $\nu(BAL_{pos}) = 0.4$, $\nu(BAL_{neg}) = 0.6$ and $\nu(PRC_{pos}) = 0.95$, $\nu(PRC_{neg}) = 0.05$. These combine to make four possible choices total, defined as $\nu(B) = \prod_{b \in B} \nu(b)$, shown in table 5.1

A description logic knowledge base L encoding the knowledge retrieved by the agent in the example may contain the following axioms, expressed in $\mathcal{SROIQ}$(D) [4]:

B	Total choice	$\nu(B)$
$B1$	BAL_{pos}, PRC_{pos}	0.38
$B2$	BAL_{pos}, PRC_{neg}	0.02
$B3$	BAL_{neg}, PRC_{pos}	0.57
$B4$	BAL_{neg}, PRC_{neg}	0.03

Table 5.1: Total choice space and their probabilities for one alignment.

$$HotelRoom \sqsubseteq Lodging, \tag{5.3}$$

$$\begin{aligned} Lodging \equiv \exists hasPrice.(xsd:integer) \sqcap \exists hasStars.(xsd:integer) \\ \sqcap \exists hasReviews.(xsd:float) \sqcap \geq 0\ \exists hasBalcony.\top, \end{aligned} \tag{5.4}$$

$$\begin{aligned} HolidayInn : HotelRoom \sqcap \exists pricePerNight.\text{"}155\text{"} \sqcap \exists hasStars.\text{"}4\text{"} \\ \sqcap \exists hasReviews.\text{"}8.3\text{"} \sqcap \exists hasFrenchBalcony, \end{aligned} \tag{5.5}$$

$$\begin{aligned} Scandic : HotelRoom \sqcap \exists hasPrice.\text{"}160\text{"} \sqcap \exists hasStars.\text{"}4\text{"} \\ \sqcap \exists hasReviews.\text{"}8.5\text{"} \sqcap \exists hasBalcony. \end{aligned} \tag{5.6}$$

Here the notation of literals is abbreviated property."a" for brevity. The full XML-based syntax would be property."a"^^xsd:integer. "The query "High quality room with balcony for no more than about 150€" is encoded by the following fuzzy dl-rules P:

$$\begin{aligned} query(x) \leftarrow^{1}_{\otimes} HotelRoom(x) \otimes HighQuality(x) \otimes \\ hasBalcony(x) \otimes hasPrice(x,y) \otimes DL[AtMostAbout150](y), \end{aligned} \tag{5.7}$$

$$\begin{aligned} HighQuality(x) \leftarrow^{0.5}_{\otimes} hasStars(x,y_1) \otimes hasReviews(x,y_2) \\ \otimes DL[AtLeast4](y_1) \otimes DL[AtLeastAbout8](y_2), \end{aligned} \tag{5.8}$$

$$hasBalcony(x) \leftarrow^{0.8}_{\otimes} DL[hasFrenchBalcony](x) \otimes BAL_{pos}, \tag{5.9}$$

$$hasPrice(x, y) \leftarrow^{0.8}_{\otimes} DL[pricePerNight](x, y) \otimes PRC_{pos}. \tag{5.10}$$

Here, $DL[AtMostAbout150](y)$ denotes the fuzzy membership of y to a fuzzy membership function $ls(150.175)$. Similarly, $AtLeastAbout8$ denotes $rs(7.5, 8.5)$ and $AtLeast4$ assigns 0.7 if $y = 4$ and 1.0 if $y = 5$, 0 otherwise. Note that the example assumes the star ratings and review scores come from a trusted source and have a probability of 1.0 of being correct. Our example, as well as the example in [25] uses min conjunction $x \otimes y = min(x, y)$. 5.7 is the user query, 5.8 encodes the concept of high quality and 5.9-5.10 are automatically generated mapping rules that encode the probabilities of correct mapping.

A fuzzy interpretation I maps a ground formula ϕ to $[0, 1]$. Each fuzzy interpretation has a probability $Pr(I)$ to be the correct one. A probability constraint $\phi \circ r$ will be

$$Pr(\phi \ \circ \ r) = \sum_{I \in \mathcal{I}, I(\phi) \circ r} Pr(I)$$

where $\mathcal{I}$ is the set of all fuzzy interpretations, $\circ$ is an operator, e.g. $\geq$ and r is a truth value $r \in [0, 1]$. Intuitively, sum up all probabilities of fuzzy interpretations I that satisfy the constraint $\circ r$. Now as $I(\phi)$ has a degree of truth and a probability $Pr(I)$, the expected truth value of ϕ is

$$\sum_{I \in \mathcal{I}} Pr(I) \cdot I(\phi)$$

and as a consequence, a probabilistic query of the form $(\phi \geq 0.8)[0.4, 0.6]$ will be satisfied if $Pr(\phi \geq 0.8)$ is in $[0.4, 0.6]$ and a query of the form $(E[\phi])[0.4, 0.6]$ is satisfied if the expected truth value of ϕ lies in $[0.4, 0.6]$. [25]

Pr is the canonical model of a probabilistic fuzzy dl-program $KB = (L, P, C, \mu)$ iff every world $I \in \mathcal{I}$ with $Pr(I) > 0$ is the canonical model of $(L, P \cup \{p \leftarrow | p \in B\})$ for some total choice B of C such that $Pr(I) = \mu(B)$. Every KB has a unique canonical model Pr. [25]

Returning to the example, let $\mathcal{I} = \{I_1, ..., I_4\}$ and $Pr(I_i) = \mu(B_i)$. The fuzzy formulas $a_1 = HighQuality(HolidayInn)$, $a_2 = hasPrice(HolidayInn, 160)$,$a_3 = DL[AtMostAbout150](155)$, $a_4 = hasBalcony(HolidayInn)$ and $a_5 = query(HolidayInn)$ are represented in table 5.2 for each canonical model I_i. Note that since 5.9 and 5.10 require the mapping of balcony and price can be inferred from the KB, in the

Canonical model	$Pr(I)$	a_1	a_2	a_3	a_4	a_5
I_1	0.38	0.56	0.8	0.67	0.4	0.4
I_2	0.02	0.56	0	0.67	0.4	0
I_3	0.57	0.56	0.8	0.67	0	0
I_4	0.03	0.56	0	0.67	0	0

Table 5.2: Degrees of truth for fuzzy formulas $a_1, ..., a_5$ in the canonical model I_i.

negative case their membership degree is 0.

The expected truth value of $E_{Pr}[query(HolidayInn)] = \sum_{I \in \mathcal{I}} Pr(I) \cdot I(query(HolidayInn))$ $Pr(I_1) \cdot I_1(query(HolidayInn)) = 0.38 \cdot 0.4 = 0.152$. For *Scandic* the calculations would produce expected truth value of 0.312, thus ranking it higher.

5.3 Generalized Theory of Uncertainty

L.A. Zadeh proposes a much more general theory for dealing with uncertainty than either fuzzy or probabilistic approaches [32; 33]. Fundamental thesis of generalized theory of uncertainty (GTU) is that information is expressed as generalized constraints. Another important aspect is that classical bivalent logic is replaced by more general fuzzy logic. Under this framework, fuzzy logic is built in throughout the system and probabilistic information can be represented as an important special case of a generalized constraint. Another important aspect of GTU in relation to the Semantic Web is the ability to process information expressed in natural language (NL). Inability to handle NL is a major shortcoming in purely probability based systems. [33]

Much of the information expressed in natural language is based on perception rather than measurement. Take the previous sentence as an example. A measurement could be "over 75% of information expressed in natural language is perception based", which is more specific and usually unnecessary in natural language. Another example is "it is very cold" versus "it is -30 degrees celsius". Fuzzy logic and more specifically linguistic variables are perfectly suited to processing this vague perception based information. [13]

Computation in GTU is done by precisiating natural language (PNL), performed by the precisiation module of a natural language processing system, as proposed by [33] (see figure 5.1). The inputs to the system are p, a system of propositions expressed in NL (INL) and q, a query also expressed in NL (QNL). The task is to compute the answer to the query $ans(q|p)$. Precisiation module precisiates the NL terms, resulting in p^* and q^*. These precisiated forms are given to the protoform module, which constructs abstracted summaries called protoforms p^{**} and q^{**}. The computation/deduction module is a knowledge base of deduction rules that govern

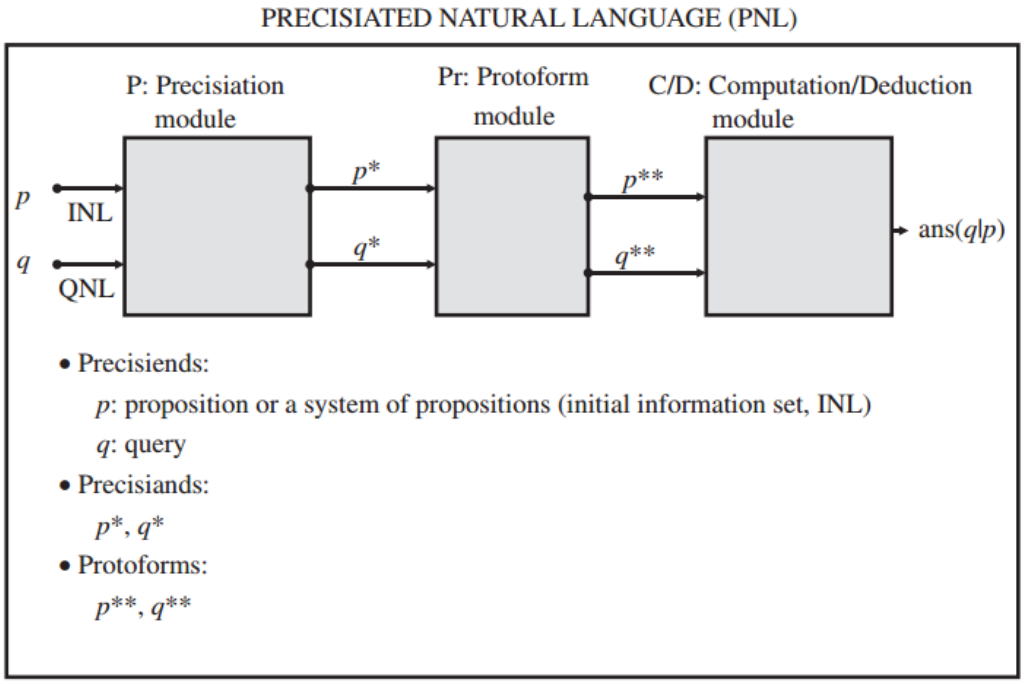

Figure 5.1: Structure of NL computation system [33].

generalized constraint propagation. [32; 33]

5.3.1 Generalized Constraint

In GTU information is expressed as generalized constraints. The meaning of proposition p, $M(p)$, carries information that places a constraint on value of X: $M(p) = GC(X(p))$. An example proposition "it is cold in Tampere" could be expressed as a constraint on the value of outside temperature, for example "Temp(Tampere) is $<$ 5". Primary generalized constraints are possibilistic, probabilistic and veristic. Other constraints can be constructed using the primary constraints. A possibilistic constraint is of the form "X is R", meaning R is the possible set of values for X. A probabilistic constraint "X isp R" means R is the probability distribution of X. A veristic constraint is of the form "X isv R", where R is the truth distribution of X. For example a mule is half horse, half donkey, or "Species(Mule) isv 0.5|Horse + 0.5|Donkey". [33]

GTU outlines a generalized constraint language (GCL), which allows semantic rules to be constructed from generalized constraints. An example of a semantic rule relating to our booking agent example is $(stars \text{ is } high) \wedge (reviews \text{ is } excellent)$ to express the concept of high quality. According to [33] this would be computed as $\text{Poss}(X \text{ is } high) \wedge \text{Poss}(Y \text{ is } excellent) = \mu_{high}(u) \otimes \mu_{excellent}(v)$, where $\otimes$ is a fuzzy conjunction. GTU does not, at present, give a complete formalism for the GCL -

merely outlines the idea of it.

With GCL it is possible to construct composite generalized constraints from the primary ones. For example a random-set constraint can be viewed as a conjunction of probabilistic constraint and either possibilistic or veristic constraint [33]. Of special interest to our question of combining fuzzy and probabilistic methods is the group of composite constraints known as bimodal distributions. The basic bimodal distribution constraint is denoted by X isbm R, where R is a bimodal distribution of the form

$$R : \sum_i P_i \backslash A_i, i = 1, ..., n. \tag{5.11}$$

Here Prob(X is A_i) is P_i, or P_i is the granular value of Prob(X is A_i). The concept of granularity is explained in subsection 5.3.2.

Essentially bimodal distribution is uncertainty about uncertainty. In practice we rarely know probabilities exactly. In information retrieval, the relevance of information is typically not known, but can be expressed as "likely to be high", for example. The basic bimodal distribution is probability on probability, see figure 5.3. Two important bimodal distributions related to the research topic are so called type 1 (possibility/probability) and type 2 (probability/possibility) bimodal distributions, see figure 5.2. The basic probability/probability bimodal distribution is a special case of type 1. [33]

Type 1 and 2 bimodal distributions have a common framework shown in equation 5.11, but differ in details of what the symbols are. In type 1, X is a random variable taking values in universe of discourse U, $A_1, ..., A_n$ are fuzzy events (fuzzy sets), p_i =Prob(X is A_i), $i = 1, ..., n$, $\sum_i p_i$ is unconstrained and P_i is the granular value of p_i. The type 1 models the granular probability P of A, which is fuzzy-set-valued. [33]

Type 2 is otherwise similar, but X is a fuzzy-set-valued random variable and the sum is constrained: $\sum_i p_i = 1$. Type 2 is denoted X isrs R, where R is the same as in equation 5.11, but with the sum constrained to one. In type 2 P is not definable, but a) the expected value of the conditional possibility of A and b) the expected value of conditional necessity of A are [33].

An example of a type 1 bimodal distribution, take the constraint "(X is *small*) isp *likely*", which means the probability of the fuzzy event "(X is *small*)" is *likely*. More formally, if X takes values in the interval $[a, b]$ and g is the probability density function of X (see figure 5.2), then

$$\text{Prob}(X \text{ is } small) = \int_a^b \mu_{small}(u) g(u) du$$

where u is a generic value of X and μ_{small} is the membership function of small. The

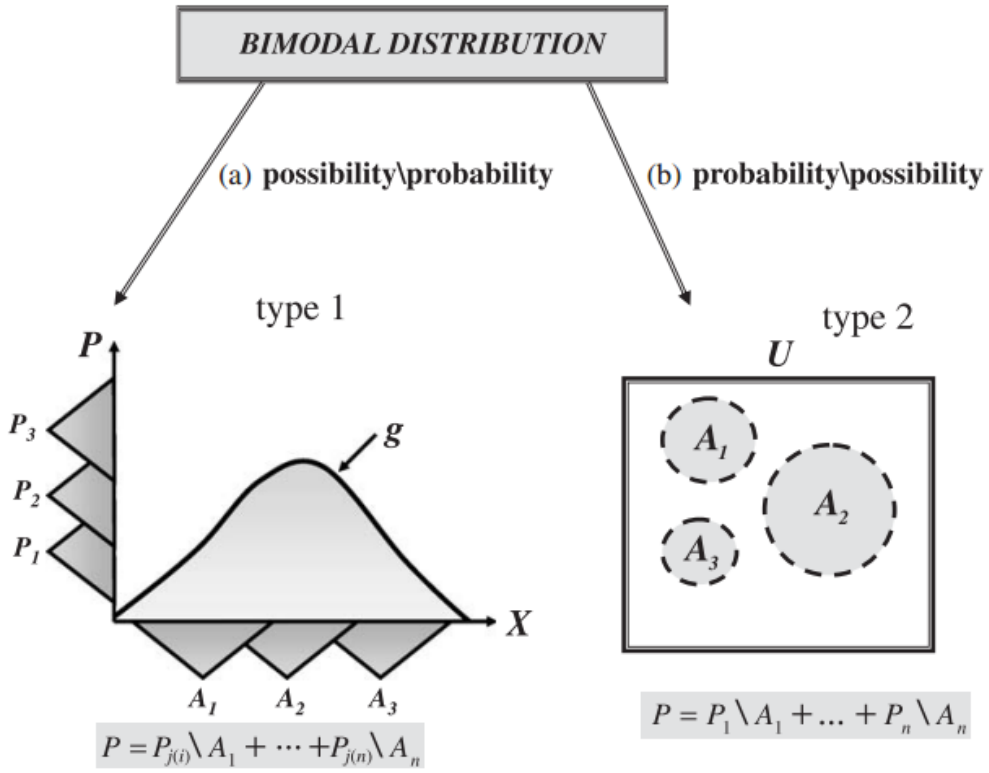

Figure 5.2: Type 1 and 2 bimodal distributions [33].

possibilistic proposition "X is *small*" is defined by

$$\text{Poss}\{X = u\} = \mu_{small}(u)$$

5.3.2 Precisiation

The process of precisiation is turning vaguely defined information into precisely defined. For example, proposition "X is approximately a", where a is a real number could be precisiated as "$a - 2 < X < a + 2$", which is a crisp-granular (cg) precisiation. Singleton precisiation, or s-precisiation, is commonly used in probability theory [32]. Granular precisiations assigns a granular value to the precisiand, defined by a generalized constraint. Forms of precisiation are singleton, crisp-granular, probability distribution, possibility distribution, fuzzy graph and bimodal distribution, depicted in figure 5.4. The aim of precisiation is to preserve the meaning of the proposition. Precisiation can be likened to a mathematical model describing a real physical system where the mathematical model is an approximation but sufficiently accurate. [33] Each precisiation of a proposition is context sensitive, and is close to the problem of finding fuzzy membership functions.

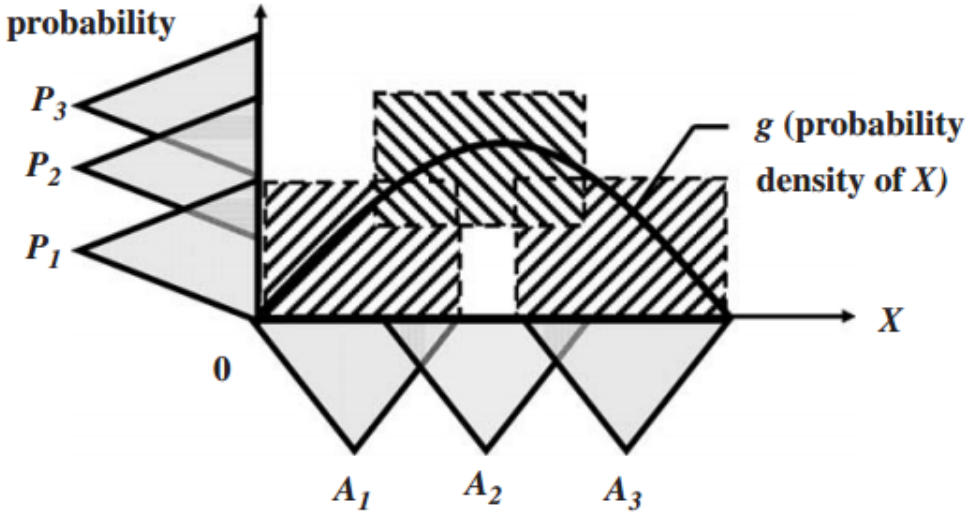

Figure 5.3: Probability/probability bimodal distribution [33].

The concept of granular value needs some explanation. Let X be a variable and a a singular value, both are in an universe of discourse U. If X is known to be a then a is the singular value of X. If there is uncertainty about the value of X then GTU expresses it in terms of a general restriction on X, X isr R, where isr is some generalized constraint. As an example, the granular value of a singular "probability is 0.9" could be "probability is high". Here 0.9 is a singular value a of X and "high" is the granular value A of X. [33]

5.3.3 Protoform

The protoform module takes the precisiated forms $p*$ and $q*$ and abstracts the variable names. For example "Alice is young" could be precisiated as "Age(Alice) is young". The protoform module replaces specific names with symbols, e.g. "A(B) is C". This prototypical form is then used in the computation/deduction module to find a correct deduction rule associated with the protoform. Note that an object may have many protoforms and many objects may have the same protoform, just as there are many possible levels of abstraction. [32; 33]

Figure 5.6 depicts the translation from object space to protoform space. Here, $S(p)$ stands for summary, or precisiated form of p, and $PF(p)$ is the abstracted summary, or protoform of p. In the example p = "Alice is young", the precisiated form becomes p^* = "Age(Alice) is young". "Age" is abstracted to a symbol A, "Alice" as B and "young" as C, thus $PF(p) = A(B)$ is C. [33]

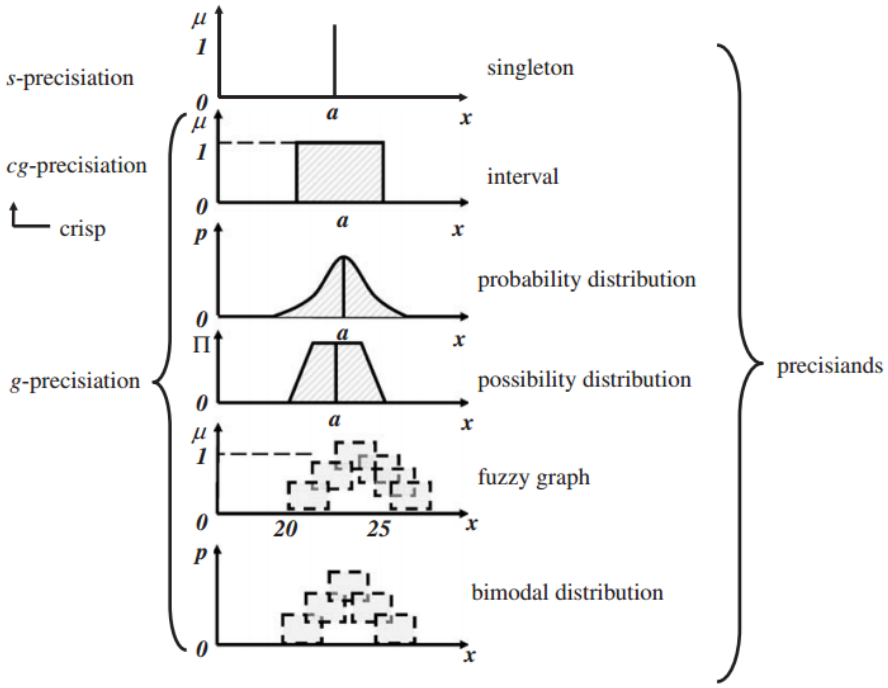

Figure 5.4: Precisiation of $^*a$ [33].

5.3.4 Computation/Deduction

The computation/deduction (C/D) module takes in the abstracted prototypical forms from the protoform module and applies a set of deduction rules to them. The C/D module is a database of deduction rules that govern constraint propagation. They are grouped into modules, depicted in figure 5.7, where each module comprises of rules for corresponding class of generalized constraints. The world knowledge module is a part of the system that incorporates domain knowledge, similar to what ontologies do in the traditional semantic web.

The rules have two parts, a symbolic protoformal part which is used to match protoformal propositions. The second part of a rule is the computational part, which specifies how the associated protoform should be processed. An example of a rule is the probability rule [33]:

$$\begin{array}{l}\text{Symbolic}\\ \dfrac{\text{Prob}(X \text{ is } A) \text{ is } B}{\text{Prob}(X \text{ is } C) \text{ is } D}\end{array}$$

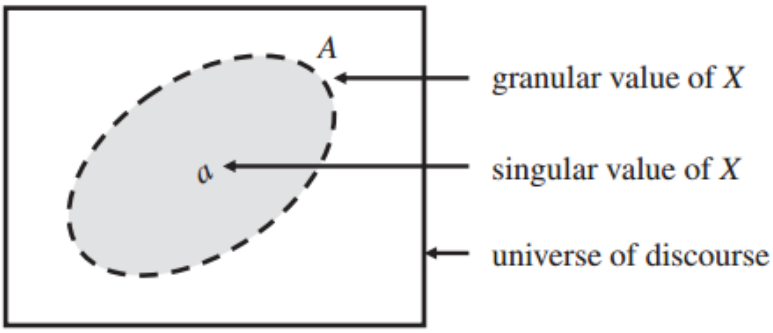

Figure 5.5: Concept of granularity [33].

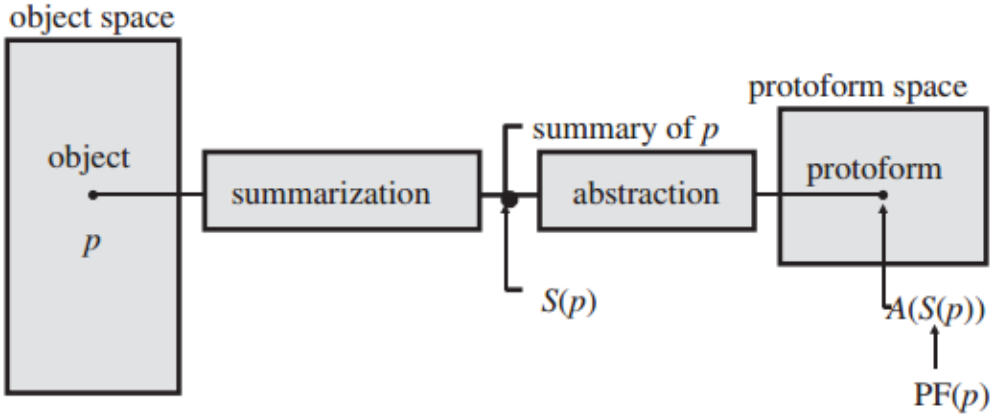

Figure 5.6: Definition of protoform of p [33].

Computational

$$\mu_D(v) = \sup_r(\mu_B(\int_U \mu_A(u)r(u)du)),$$

$$\text{subject to } v = \int_U \mu_C(u)r(u)du,$$

$$\int_U r(u)du = 1$$

where X is a real-valued random variable, A, B, C and D are fuzzy sets, r is the probability density of X and $U = \{u\}$. [33] presents an outline of a C/D rulebase consisting of ten principal deduction rules in symbolic form.

Zadeh [33] gives several examples on computing with generalized constraints. Calculation in GTU is viewed as query answering. Consider a situation where proposition is p = "Most Swedes are tall" and the question q = "What is the average height of Swedes?". The precisiated forms can be expressed as

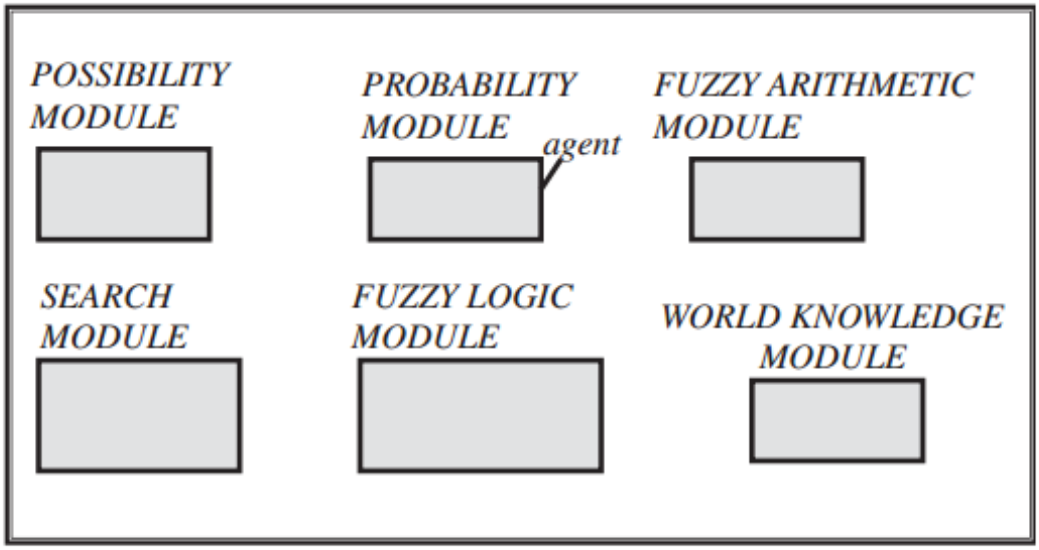

Figure 5.7: Computation/deduction module [33].

$$p^* : \frac{1}{n}(\mu_{tall}(h_1) + \cdots + \mu_{tall}(h_n)) \text{ is most},$$
$$q^* : \text{ans}(q|p) = \frac{1}{n}(h_1 + \cdots + h_n).$$

where h_i is the height of i:th Swede in population $G = (u_i, ..., u_n)$ of Swedes, $\mu_{tall}(h)$ is the membership function of "tall". Let $\mu_{ave}(v)$ be the membership function of average height. By applying the extension principle rule from the database [33; 34], computation reduces to variational problem

$$\mu_{ave}(v) = \sup_h \left(\mu_{most}\left(\frac{1}{n}\left(\mu_{tall}(h_1) + \cdots + \mu_{tall}(h_n)\right)\right)\right)$$

Another example given in [32] is the balls-in-box problem: A box contains about 20 black and white balls. Most are black. There are several times as many black balls as white. What is the number of white balls? In precisiated form:

$$p_1 : (X + Y) \text{ is } {}^*20$$
$$p_2 : X \text{ is most} \times {}^*20$$
$$p_3 : X \text{ is several } \times Y$$
$$q : Y \text{ is } ?A$$

where ^{*20} is approximately 20 and $?A$ denotes the queried variable. Approxi-

mately $^*a$ is a problem of precisiation, i.e. is a generalized constraint on the value a that needs to be precisiated suitably. [32] does not give detailed explanation how to calculate the example, merely states that it reduces to fuzzy integer programming, see figure 5.8.

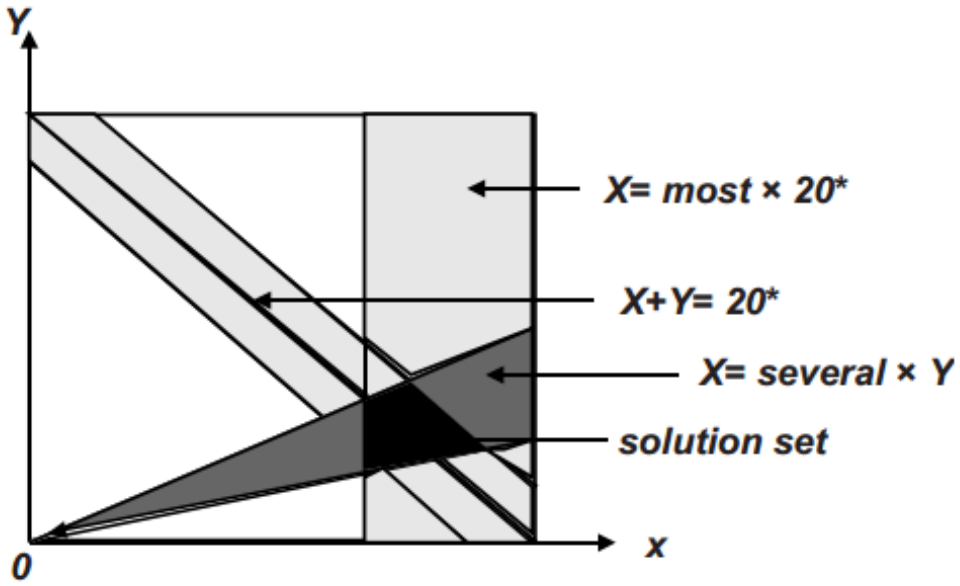

Figure 5.8: Balls-in-box problem solution [32].

5.3.5 Booking agent example

Expressing the booking agent example in terms of GTU, let proposition p be "HolidayInn has good reviews, probably doesn't have a balcony and costs 155€. Scandic has excellent reviews, probably has a balcony and costs 160€." and the question q "Which hotel ranks higher, considering reviews, balcony and price?". The precisiated forms could be

$$\begin{aligned} p_1* = \text{Reviews(HolidayInn) is } &\textit{good}, \\ \text{Balcony(HolidayInn) isp } &\textit{unlikely}, \\ \text{Price(HolidayInn) is } &155. \end{aligned} \tag{5.12}$$

$$\begin{aligned} p_2* = \text{Reviews(Scandic) is } &\textit{excellent}, \\ \text{Balcony(Scandic) isp } &\textit{likely}, \\ \text{Price(Scandic) is } &160. \end{aligned} \tag{5.13}$$

$$q* = \text{Reviews}(?X) \text{ is } high \wedge \text{Balcony}(?X) \text{ is } likely \wedge \text{Price}(?X) \text{ is } low \quad (5.14)$$

The GTU, as presented in [32; 33] is unfortunately not mature enough to calculate this example without developing the theory further. According to the principles of GCL, the query could be formulated as the conjunction of the constraints on Reviews, Balcony and Price. Unfortunately, GTU does not present strict formalisms to do this. There are three major areas where GTU should be developed further for it to be implementable. 1) Present complete formalism for the generalized constraint language , or, as [33] notes this is an infinite set, a subset of the GCL called the standard constraint language (SCL) which is likely to be enough for practical applications. 2) Present methods to precisiate any natural language proposition. 3) Develop algorithms for the precisiation, protoform transformation and computation/deduction.

A possible way of calculating the example might be to conjunctively combine the terms in proposition p_1* to get "(Reviews(HolidayInn) is *good*) $\wedge$ (Balcony(HolidayInn) isp *likely*) $\wedge$ (Price(HolidayInn) is 155)". To have a semantic meaning, this proposition needs a GCL, for example how to conjunctively combine *good* and *likely*. One is a fuzzy membership degree, the other is a probability distribution. One interpretation for this could be borrowed from Probabilistic Fuzzy Description Logic Programs explained in section 5.2 - the conjunction of a fuzzy membership degree with probability is the expected truth value. It is not immediately clear how this could be used in GC propagation in the C/D module. A simple rule could possibly reduce the calculation to the application of the rules presented in [33], but more research is required to ascertain how.

The generalized theory of uncertainty is by no means a mature technology, but might offer semantic web agents a way to reason with uncertainties with different semantics - vague statements about incomplete information or probabilistic statements about vague information. That is, fuzzy statements on probabilistic information or probabilistic statements on fuzzy information. E.g. "There is heavy snowfall in Tampere", with the snowfall being a probability distribution and heavy a fuzzy membership. Conversely, "Bob is likely to be tall", where Bob being tall is a fuzzy membership and likely a probability distribution on the fuzzy statement.

6. CONCLUSION

Describing information in a way that is accessible to both humans and machines is no trivial task. There are challenges in developing the theoretical back-end of logics for ontologies so that machines can reason with semantic information. Practical algorithms need to be decidable - terminate in a reasonable time. Additionally, the Semantic Web brings a challenging environment where information can be untrustworthy, missing or vague. Some of the most prominent approaches in the literature in dealing with uncertainty were described.

W3C Uncertainty Reasoning for the World Wide Web Incubator Group divides uncertainty into six types: ambiguity, empirical, randomness, vagueness, inconsistency and incompleteness. They can be seen as special cases of two types of information: information for which definition is lacking (vague, inconsistent, ambiguous) and incomplete information (empirical, randomness, incompleteness). Roughly fuzzy methods are best suited for vague information while probabilistic methods are best for incomplete information.

Probabilistic approaches have a strong theoretical background and long tradition in dealing with uncertainty across all fields of research. Applied to the Semantic Web, the most prominent probabilistic method is MEBN/PR-OWL. It combines first order probabilistic logic with Semantic Web ontologies. The downside is defining uncertainty in terms of MEBN is a tedious, error prone task which requires deep knowledge of both the application domain and MEBN theory.

One of the selling points of fuzzy logic is that it is easy to interpret. Applied to the Semantic Web, Fuzzy OWL 2 provides a simple way to add fuzzy annotations to OWL 2, using existing OWL 2 syntax and editors. For fuzzy reasoning, dedicated fuzzy reasoning systems need to be used, such as FiRE, DeLorean or FuzzyDL.

The main emphasis was to consider situations where several semantically different forms of uncertainty occur in a single application. A hotel booking agent example was provided as a way to illustrate the sources of uncertainty and the approaches in dealing with them. The probabilistic fuzzy description logic program approach by Straccia and Lukasiewicz [25] provides a ready to implement theory with considerable thought put into algorithms and implementation.

Fuzzy description logics are, however, less flexible than the general theory of uncertainty proposed by Zadeh [33]. GTU is a much less mature theory, providing no

strict formalisms to translate every NL proposition into the generalized constraint form and much less in the way of decidable algorithms for processing and reasoning. It does not consider large knowledge bases and the efficiency of a possible implementation either. It would require 1) strict formalism of GCL, 2) a detailed method to precisiate any NL proposition and 3) efficient algorithms.

In theory, a unified framework for dealing with all kinds of uncertainty would be ideal. GTU, at present, is not complete or ready for implementation. Probabilistic fuzzy description logic program approach is, and will likely perform sufficiently in practice. Like the Semantic Web, the future of it is uncertain. Which methods will see widespread use, only time will tell, but one would think the methods that are sufficiently effective in practice and require the least effort to implement and maintain should prevail.

REFERENCES

[1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American Magazine, May 2001, pp. 29-37.

[2] J. Domingue, D. Fensel, J. A. Hendler, Handbook of Semantic Web Technologies, Introduction to the Semantic Web Technologies, Springer, Heidelberg, Germany, 2011, pp. 1-41.

[3] G. Antoniou, F. van Harlemen, A Semantic Web Primer, 2nd edition, The MIT Press, Cambridge, Massachusetts, USA, 2008, 225 p.

[4] Straccia, U. Foundations of Fuzzy DL and Semantic Web, CRC Press, Boca Raton, Florida, USA, 2014, 320 p.

[5] I. Mozetic, W3C RIF-WG Wiki, Horn Logic, available (referenced 19.11.2015): `http://www.w3.org/2005/rules/wg/wiki/Horn_Logic`.

[6] F. Baader, U. Sattler, An Overview of Tableau Algorithms for Description Logics, Studia Logica, Vol. 69, No. 1, 2001, pp. 5-40.

[7] K. J. Laskey, K. B. Laskey, P. C. G. Costa, M. M. Kokar, T. Martin, T. Lukasiewicz, Uncertainty Reasoning for the World Wide Web, W3C Incubator Group Report 31 March 2008, available (referenced 19.11.2015): `http://www.w3.org/2005/Incubator/urw3/XGR-urw3/`.

[8] Unceritainty upper ontology by the W3C Uncertainty Reasoning for the World Wide Web Incubator Group, available (referenced 19.11.2015): `http://www.w3.org/2005/Incubator/urw3/XGR-urw3/Uncertainty.owl`.

[9] M. Sabou, M. d'Aquin, E. Motta, Exploring the Semantic Web as Background Knowledge for Ontology Matching, in: Journal on Data Semantics XI, Vol. 5383, Springer, Heidelberg, Germany, 2008, pp. 156-190.

[10] C. d'Amato, Similarity-based Learning Methods for the Semantic Web, PhD Thesis, 2007, available (referenced 19.11.2015): `http://www.di.uniba.it/~cdamato/PhDThesis_dAmato.pdf`.

[11] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the Semantic Web, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 6, No. 4, 2008, pp 291-308.

[12] D. Dubois, H. Prade, Possibility theory, probability theory and multiple-valued logics: a clarification, Annals of Mathematics and Artificial Intelligence, Vol. 32, No. 1, 2001, pp. 35-66.

[13] L. A. Zadeh, Is there a need for fuzzy logic?, Information Sciences: an International Journal, Vol. 178, No. 13, 2008, pp. 2751-2779.

[14] T. J. Ross, Fuzzy Logic with Engineering Applications, 3rd Edition, Wiley, 2010, 578 p.

[15] F. Baader, R. Peñaloza, GCIs Make Reasoning in Fuzzy DL with the Product T-norm Undecidable, Proceedings of the 24th International Workshop on Description Logics (DL2011), Vol. 745, 2011.

[16] F. Bobillo, U. Straccia, Syntax and Semantics of FuzzyDL, available (referenced 19.11.2015): `http://www.umbertostraccia.it/cs/software/fuzzyDL/download/old/documents/syntax.pdf`.

[17] F. Bobillo, M. Delgado, J. Gómez-Romero, DeLorean: A reasoner for fuzzy OWL 2, Expert Systems with Applications, Vol. 39, No. 1, 2012, pp 258-272.

[18] N. Simou, G. Stoilos, G. Stamou, Storing and Querying Fuzzy Knowledge in the Semantic Web Using FiRE, Uncertainty Reasoning for the Semantic Web II, Springer, Heidelberg, Germany, 2013, pp 158-176.

[19] F. Bobillo, U. Straccia. Fuzzy ontology representation using OWL 2. International Journal of Approximate Reasoning, Vol. 52, No. 7, 2011, pp.1073-1094.

[20] F. Bobillo, U. Straccia, Fuzzy Ontology Representation using OWL 2, available (referenced 19.11.2015): `http://www.umbertostraccia.it/cs/software/FuzzyOWL/`.

[21] P. C. G. da Costa, K. B. Laskey, K. J. Laskey, PR-OWL: A Bayesian Ontology Language for the Semantic Web, Uncertainty Reasoning for the Semantic Web I, Springer, Heidelberg, Germany, 2008, pp. 88-107.

[22] Z. Ding, Y. Peng, A Probabilistic Extension to Ontology Language OWL, Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Vol. 4, No. 4, 2004.

[23] T. Gu, H. K. Pung, D. Q. Zhang, A Bayesian Approach For Dealing With Uncertain Contexts, Austrian Computer Society, 2004.

[24] R. Giugno, T. Lukasiewicz, P-SHOQ(D): A Probabilistic Extension of SHOQ(D) for Probabilistic Ontologies in the Semantic Web, 2002.

[25] T. Lukasiewicz, U. Straccia, Description logic programs under probabilistic uncertainty and fuzzy vagueness, International Journal of Approximate Reasoning, Vol. 50, No. 6, 2009, pp. 837-853.

[26] T. Koski, J. M. Noble, Bayesian Networks: An Introduction, Wiley, 2009, 330 p.

[27] R. Carvalho, K. Laskey, L. Santos, M. Ladeira, P. Costa, S. Matsumoto, UnBBayes: Modeling Uncertainty for Plausible Reasoning in the Semantic Web, INTECH Open Access Publisher, 2010.

[28] G. Shafer, The Construction of Probability Arguments. Boston University Law Review, Vol. 66, 1986, pp. 799-823.

[29] R. N. Carvalho, K. B. Laskey, P. C. G. Costa, PR-OWL 2.0 - Bridging the Gap to OWL Semantics, Uncertainty Reasoning for the Semantic Web II, Uncertainty Reasoning for the Semantic Web II, Springer, Heidelberg, Germany, 2013, pp.1-18.

[30] J. Callan, Distributed information retrieval, In: W.B. Croft (Ed.), Advances in Information Retrieval, Kluwer Academic Publishers, 2000, pp. 127-150.

[31] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, New York, USA, 2007.

[32] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU) - an outline, Information Sciences, Vol. 172, 2005, pp. 1-40.

[33] L. A. Zadeh, Generalized theory of uncertainty (GTU) - principal concepts and ideas, Computational Statistics & Data Analysis, Vol. 51, No. 1, 2006, pp 15-46.

[34] L. A. Zadeh, Fuzzy sets, Information and Control, Vol. 8, No. 3, 1965, pp 338-353.