UNIVERSITY OF SOUTHAMPTON

Faculty of Physical and Applied Sciences

School of Electronics and Computer Science

N-Quads: A Bayesian extension for the Semantic Web

by

Keyvan Mir Mohammad Sadeghi

A dissertation submitted in partial fulfillment of the degree of

MSc in Artificial Intelligence

by examination and dissertation

January 2012

SUPERVISOR: Dr. Nicholas M. Gibbins

ABSTRACT

One of the criticisms that has been raised against the current semantic web languages is their inability to represent uncertain knowledge (Shadbolt et al., 2006, Fensel and Van Harmelen, 2007). In the real world applications, it is often the case that the knowledge cannot be represented as simple *TRUE* and *FALSE* values. In addition, there are other scenarios in which the 'trueness' of the data is questionable, e.g. data that is coming from an untrusted source. The nature of uncertainty makes the extent to which we are certain about some piece of knowledge to depend on the certainty about some other pieces of knowledge. Current standards of the Semantic Web are not capable of representing/manipulating uncertainty. The objective of this work is to address this issue by proposing an extension for the Semantic Web to fill this gap.

ACKNOWLEDGEMENT

I would like to thank Dr. Nicholas M. Gibbins for his kindness and patience. If it was not for his guidelines, this dissertation would not be anything like what it now is. In addition, I have to thank Scott Doswell for his dedication in proof reading.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The roots of many criticisms (Shirky, 2003, Floridi, 2009) that have been raised against the Semantic Web can be traced back to the initial failure in the field of Artificial Intelligence. Success in developing solutions for specific domains during early years of research (e.g. an AI system that defeated the chess champion) led the AI researchers to give optimistic promises about the time of achieving human-like intelligence. Some even went as far as saying that they have solved the hard problem of AI, for example, Herbert Simon once said that "there are now in the world machines that think, that learn and that create" (Simon and Newell, 1958). In 1999, Berners-Lee gave a similar promise stating that by the time that we have the Semantic Web ready, "the intelligent agents people have touted for ages will finally materialize" (Berners-Lee, 1999). The critics say that by analogy, this is not a promise that could be kept. However, we can learn from the AI experience with the aims of pushing the Semantic Web towards the right direction by identifying the weak points and corresponding solutions that have been put forward. For Instance, some argue that particular shifts in direction such as uncertainty handling could have substantial impacts on the AI (Korb and Nicholson, 2004). We claim that the same argument holds for the Semantic Web.

We will start by presenting our research question. What is the main detail that has been missed in the current proposal of the Semantic Web, which if addressed effectively, the result would be a dramatic increase in usefulness of the Semantic Web? This question is rather broad and a broad answer cannot be evidently supported. Accordingly, we should narrow our question down. Flexible knowledge management is an obvious requirement for the future of the Semantic Web. Within the scope of flexibility, research has shown that there is an important feature that is missing, ability to handle uncertainty (Shadbolt et al., 2006, Fensel and Van Harmelen, 2007). Various methods have been investigated on how uncertainty could be handled in the Semantic Web, e.g. probabilistic (Lukasiewicz, 2007) and Fuzzy (Straccia, 2011) extensions . From all those, we decided to go along with the Bayesian theory, since it can be supported by extensive mathematical background and has proven to be effective and useful. Therefore, we can summarise our question as follows:

Research Question: How can Bayesian Networks best be integrated with the Semantic Web to provide uncertainty handling?

Current Semantic Web languages (i.e. RDF, RDF Schema and OWL) are equipped with mechanisms to represent the dependencies in between one piece of knowledge to another. However, they lack the feature of representing the uncertainty within those dependencies because of their design. This work tries to address this issue by proposing an extension for the Semantic Web, a new language, to enrich it with the ability to represent uncertain knowledge. The goal is to provide a straightforward way of representing uncertainty by combining the elements that bring about the uncertainty in the first place, dependencies and probabilities. To achieve this, we will utilise Bayesian Networks, a probabilistic framework that represents probability in context of dependencies. In order to prove the hypothesis, we need to build a reasoner that enables us to reason about the uncertain knowledge encoded by this method.

Here is how this dissertation is structured:

Section 2 reviews the literature relating to the work that we want to do.

Section 3 will analyse the requirements of this work.

Section 4 describes the design and implementation of the extension.

Section 0 evaluates the usability and effectiveness of the model.

Section 6 concludes the dissertation and suggests directions for future work.

2 LITERATURE REVIEW

In this section, we review the background relating to this work. We do so by starting with a description of the traditional approaches of knowledge representation and reviewing the methods that have been investigated in past for handling uncertainty within the scope of knowledge management. Section 0 changes the focus to the Semantic Web by describing what the web is, and what had been envisioned to reach for the web with semantics that would not be possible to reach without them. We then continue to the technologies that have been supplied by the World Wide Web consortium for encoding knowledge in the Semantic Web. In the latter section, the methods of establishing uncertainty in the Semantic Web will be discussed and the relating research will be reviewed.

2.1 KNOWLEDGE REPRESENTATION

Reasoning based on logic can be traced back to Aristotle's syllogisms, a restricted form of argument that allows a proposition to be inferred from a given set of premises. Both proposition's structure and reasoning within this process are defined in an *ad hoc* basis. Moreover, the knowledge is encoded in form of facts and rules, each fact can only have the value of either *TRUE* or *FALSE* and each rule defines what conclusion is drawn from the given premises. Consequently, the process of reasoning will be entirely deterministic and there will be a unique solution in each syllogism. From the 1960s, a substantial effort started to import the traditional inference methods to the computers. *Semantic Networks* have been developed on account of work by Quillian in which he introduced a framework for drawing conclusion from English text (Quillian, 1967, Collins and Quillian, 1969). He outlined that the model is to be "the base of knowledge underlying human-like language behaviour". The same motives led Minsky to work on another structure type known as the *frame systems* (Minsky, 1974). There are many types of knowledge representation proposed by others such as Conceptual Graphs (Sowa, 1983), Conceptual Dependency Theory (Schank, 1972) and Scripts (Schank and Abelson, 1977). These models have been adopted in many domains and proven to be effective in terms of both representation and reasoning.

2.1.1 FRAME LOGIC

Frames are data structures, designed to characterise the specifications of a particular object in a knowledge domain. Frame Systems take the relational logic of Semantic Networks and extend it with hierarchical associations. We have two types of frame, 'Class Frame' and 'Instance Frame'. A frame has a number of 'Slots', each can point to other frames or slots. We also have the notion of 'demon' in frames, that is, a procedure associated with a slot that fires every time the slot is accessed. Frames are the basic concept behind 'Object Orientation' paradigm. Accordingly, we can identify a mapping to OO principles that a frame is identical to a class, a slot is a field or a method, and demons are properties.

$$\left.\begin{array}{l} \forall\, Intelligent(Student) \Rightarrow\ Successful(Student) \\ \forall\, Successful(Student) \Rightarrow\ Progresses(Student, fast) \\ Intelligent(peter) \end{array}\right| \Rightarrow \begin{array}{l} Successful(peter) \\ Progresses(peter, fast) \end{array}$$

FIGURE 1: REASONING IN FRAME LOGIC

Frames also utilise the concept of 'inheritance' which allows defining sub classes of a frame that inherit some of the specifications of the super class and possibly override some others. This is the key idea of the frame logic that relieves us from having to specify redundant characteristics in a hierarchy. Figure 2 shows an example where knowledge can be organised in hierarchical structure.

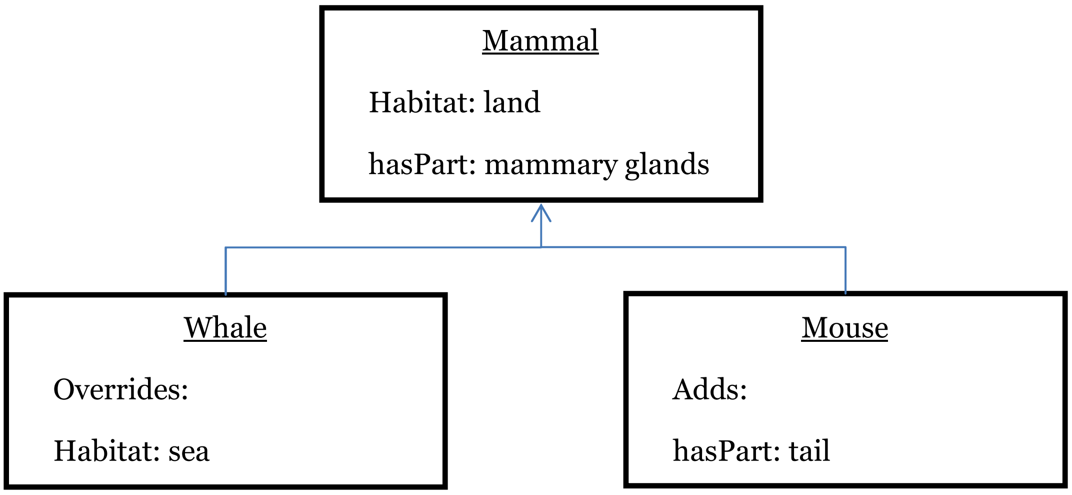


FIGURE 2: KNOWLEDGE HIERARCHY IN FRAME SYSTEMS

2.1.2 DESCRIPTION LOGIC

Although Semantic Networks and Frame Logic provided suitable structures for knowledge representation, the lack of semantic formalism caused each system to behave differently. As a result, the need for semantic integrity for hierarchical data emerged and the semantic unification has been introduced, labelled by *Description Logic (DL)*. In broad terms, DL is a subset of First-order Predicate Logic (FOPL) of limited expressivity.

Back in the 90s, many commercial projects (as well as academic ones) had been developed that had their roots in DL. Below is a list of successful projects:

- KL-ONE by (Brachman and Schmolze, 1985)
- CLASSIC by (Brachman et al., 1991)
- European GALEN project by (Rector and Nowlan, 1994)
- GRAIL by (Rector et al., 1997)

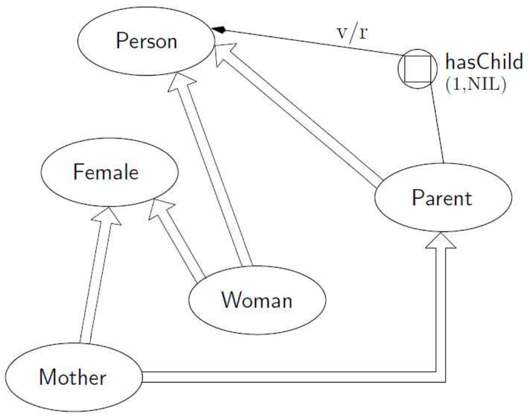


FIGURE 3: AN EXAMPLE NETWORK,
ADAPTED FROM (NARDI AND BRACHMAN, 2003)

DL has its own syntax, which is essentially the sugared syntax for FOPL. We have the notion of 'TBox', a set of axioms that let us describe knowledge about a domain. We also have DL operators, which are to a great extent analogous to the operators of the set theory. For example, we can have:

$$Woman \equiv Person \sqcap Female$$

This specifies two things. Firstly, it describes a woman as a person who is female. Secondly, this TBox indicates that the intersection of the classes of all things which are females and the class of all persons is identical to the class of women. This can be translated to First-order predicate logic as follows:

$$\forall x \; Woman(x) \Leftrightarrow Person(x) \wedge Female(x)$$

An 'ABox' is a collection of axioms which makes it possible to extend the knowledge about a specific domain. Assertions of this type are used to define instances and can be done in two ways:

1. Concept Assertion: $Female \sqcap Person(ANNA)$
2. Role Assertion: $hasChild(ANNA, JACOPO)$

We have the universal class (top), '⊤', defined as the superclass of all things and contradiction (bottom), denoted as '⊥'. One can narrow down the domain of a concept by putting universal or existential restrictions:

$$\forall hasPet.Cat \, , \exists hasPet.Cat$$

The first restriction refers to the class of all things whose pets are only cats, whereas the second restriction points at the class of all things that at least have a pet cat. We also have the number restrictions:

$$\geq 2 \; hasChild, = 4 \; hasLeg$$

The first cardinality restriction refers to the class of things who at least have two children, and the second expression points at the class of quadrupeds. By combining these axioms, we can address more specific classes:

$Man \sqcap Succesful \sqsubseteq \exists hasWife.Beautiful$	Every successful man has a beautiful wife.
$Penguin \sqcap FlyingThing \equiv \bot$	Penguins cannot fly.

$$Curry \equiv Stew \sqcap \exists hasOrigin.\{India\} \sqcap \exists hasIngredient.Spicy$$

Curry is an Indian stew which contains at least one spicy ingredient.

It is possible with the aid of ABox and TBox axioms to encode complex knowledge about a domain. This makes DL a perfect candidate for encoding knowledge. However, for practical applications, expressivity is not the only factor that should be taken into account. More important than being expressive is the time it takes for the reasoning to be small enough for knowledge to be useful. In order to differentiate levels of expressiveness, we have DL naming conventions illustrated in Table 1 and Table 2.

$\mathcal{AL}$ (Attributive language)	$\mathcal{FL}$ (Frame language)	$\mathcal{EL}$
• Atomic negation • Concept intersection • Universal restrictions • Limited existential quantification	• Concept intersection • Universal restrictions • Limited existential quantification • Role restriction	• Concept intersection • Existential restrictions

TABLE 1: NAMING CONVENTIONS FOR BASIC LOGICS

$\mathcal{F}$	Functional properties	$\mathcal{R}$	Limited complex role inclusion axioms
$\mathcal{E}$	Full existential qualification	$\mathcal{O}$	Nominals
$\mathcal{U}$	Concept union	$\mathcal{I}$	Inverse properties
$\mathcal{C}$	Complex concept negation	$\mathcal{N}$	Cardinality restrictions
$\mathcal{H}$	Role hierarchy	$\mathcal{Q}$	Qualified cardinality restrictions
$^{(\mathcal{D})}$	Data type properties, data values or data types		

TABLE 2: DL EXTENSIONS TO BASIC LOGICS

By using the naming conventions, we can identify different classes of expressiveness. For example, $\mathcal{ALC}$ is the class of expressivity that allows basic logical operations from attributive language ($\mathcal{AL}$) over the concepts (concepts are allowed as $\mathcal{C}$ extension indicates). Since $\mathcal{ALC}$ is an obvious subset of most classes of expressivity, it can be shortened to $\mathcal{S}$. Therefore, we can have levels of expressiveness such as $\mathcal{SROIQ}^{(\mathcal{D})}$, allowing operators from $\mathcal{ALC}$ (as suggested by $\mathcal{S}$) and the following extensions (namely, $\mathcal{R}$, $\mathcal{O}$, $\mathcal{I}$, $\mathcal{Q}$ and $^{(\mathcal{D})}$).

For the Semantic Web, DL provides the logical formalism needed for semantic unification across all implementations. In addition, classifying our encoded knowledge by the level of expressiveness will enable us to measure the effectiveness of reasoning. This is important since there is trade-off between expressivity and reasoning time, the more expressive the logic, the less the worst-case time complexity will be applicable. Imagine if we had a huge knowledge base with all the information we need, in case that the system did not respond in the time for it to be useful to us, the whole effort of encoding the knowledge would be useless. Therefore, the class of expressivity should always be taken into account in the process of encoding knowledge, with respect to the domain within which the knowledge is to be consumed.

2.2 KNOWLEDGE REPRESENTATION AND UNCERTAINTY

Uncertainty is the undeniable fact that surrounds every aspect of our knowledge. Since "knowledge is an unending adventure at the edge of uncertainty" (Bronowski, 1974) and "uncertainty is the only certainty there is" (Paulos, 1988), one might suggest that reasoning about uncertain knowledge is certainly an unending challenge. In fact, understanding the concept of uncertainty has been made possible through the aid of numerical representation and probabilities. However, adopting the probabilistic approach in different domains remains a challenge. This section borrows ideas mainly from two books by (Halpern, 2005) and (Russell and Norvig, 1998). We will go through mathematical definitions of uncertainty as well as the methods for handling it.

2.2.1 PROBABILISTIC FRAMEWORK

In the probabilistic paradigm, knowledge is defined in terms of events. Each *event* is a piece of knowledge with some degree of uncertainty. Let U be an event and μ be the probability measure, the probability of event U is written as $\mu(\mathrm{U})$, we have:

$$\mu(\mathrm{U}) = \frac{n_U}{\mathrm{N}}, \qquad 0 \le \mu(\mathrm{U}) \le 1, \qquad \textstyle\sum_i \mu(\mathrm{U}_i) = 1$$

Where N is the sum of all possible outcomes in a discrete domain and n_U is the share of event U within the N possibilities. It can be drawn that computing the probability of an event is possible by a simple statistical survey in discrete domains.

Conditional probability is defined as the probability of event U when given another event, say V, from the same probability space:

$$\mu(\mathrm{U}\,|V) = \frac{\mu(\mathrm{U}\ \cap \mathrm{V})}{\mu(\mathrm{V})}$$

$\mu(\mathrm{U} \mid \mathrm{V})$ is undefined when $\mu(\mathrm{V})$ = zero. By intuition, two events are independent when the occurrence of one does not affect the other. In mathematical context, given the probabilistic measure μ, events U and V are independent if and only if both of the following conditions are satisfied:

1. $\mu(\mathrm{U} \mid \mathrm{V}) = \mu(\mathrm{U})$ where $\mu(\mathrm{V}) \neq 0$
2. $\mu(\mathrm{V} \mid \mathrm{U}) = \mu(\mathrm{V})$ where $\mu(\mathrm{U}) \neq 0$

In case that either of the above conditions does not hold true, we say, events U and V are probabilistically dependent. In general, the condition for *mutual independence*, i.e. pair-wise independence for a set of events, is as follows:

$$\mu\left(\bigcap_i \mathrm{U}_i\right) = \prod_i \mathrm{U}_i$$

2.2.2 BAYESIAN NETWORKS

A Bayesian network (also known as belief network) is essentially a directed acyclic graph (DAG) with nodes labeled by random variables, designed to ease the process of working with probability measures. Nodes in a Bayesian network represent events while vertices show the dependencies in between. A Bayesian network provides the best way for visualising conditional independencies as the path from the two variables through the cause is easy to trace.

Based on the concept of conditional independence, *Bayes' Rule* is defined as follows:

$$P(cause|effect) = \frac{P(effect|cause)P(cause)}{P(effect)}$$

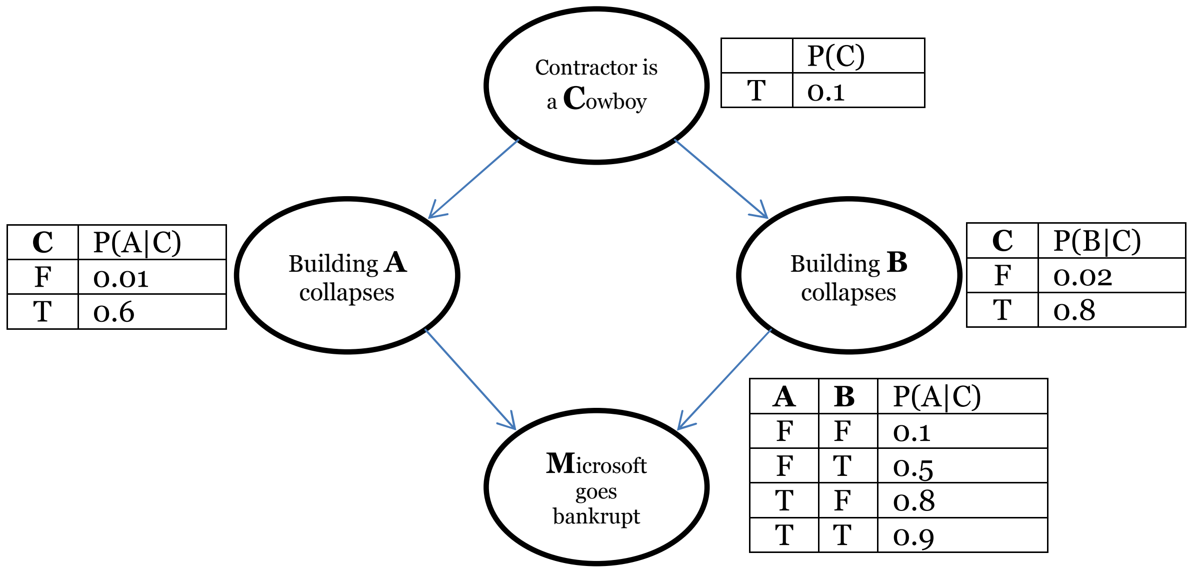
The key idea in Bayesian Theory is that one can reason about a diagnostic probability based on the causal probability. The following terms are often used in the literature:

Prior : P(cause)

Evidence : P(effect)

Likelihood: P(B|A)

Posterior Probability :P(cause|effect)



	P(C)
T	0.1

C	P(A|C)
F	0.01
T	0.6

C	P(B|C)
F	0.02
T	0.8

A	B	P(A|C)
F	F	0.1
F	T	0.5
T	F	0.8
T	T	0.9

FIGURE 4: MICROSOFT'S BUILDINGS EXAMPLE, ADAPTED FROM (NOBLE, 2007)

Figure 4 shows a Bayesian network that encapsulates the dependencies relating Microsoft's Bankruptcy, the status of two buildings and builder's loyalty. In case that the builder is a Cowboy (the term used by Englishmen for describing untrustworthiness of contractors), there is a great chance of having the buildings collapse. Meanwhile, Microsoft's wealthyness could promptly vanish by having either of the buildings collapse.

What makes Bayesian Networks important in the scope of this work is their ability of robust reasoning in a domain that contains uncertain knowledge. We show the process of reasoning by examining different scenarios that can happen in the above-mentioned example. A number of questions will be reviewed to illustrate how reasoning is done. Let 'C' be the event of contractor being a cowboy, '*A*' , building A collapses, '*B*', building B collapses and '*M*' Microsoft going bankrupt.

The topology of the network by its own can give reveal some knowledge about the domain. The first question will demonstrate reasoning without any evidence.

Question I: What can be inferred about *A* only by having the relations in the network specified?

We can answer this question by calculating $P(B)$. In the case that no evidences are provided or knowledge about of a child node is questioned, the probability of the event can be obtained by the *Normalisation* rule. That is, the sum of probabilities for all possible configurations of the network with respect to the parent nodes. Hence, we have:

$$\begin{aligned}P(A) &= P(A|C)P(C) + P(A|\overline{C})P(\overline{C})\\ &= 0.6 \times 0.1 + 0.01 \times 0.9 = 0.069\end{aligned}$$

Where $P(\overline{C})$ is calculated as $1 - P(C)$.

We can calculate $P(B)$ by a similar approach:

$$\begin{aligned}P(B) &= P(B|C)P(C) + P(B|\overline{C})P(\overline{C})\\ &= 0.8 \times 0.1 + 0.2 \times 0.9 = 0.098\end{aligned}$$

Question II: What can we infer about *M* when we know that contractor is a cowboy?

Once again, we should adapt the Normalisation Rule since the probability of a child node has been questioned. We have:

$$\begin{aligned}&P(M|C) = P(M|A \cap B)P(A)P(B) + P(M|A \cap \overline{B})P(A)P(\overline{B})\\ &+P(M|\overline{A} \cap B)P(\overline{A})P(B) + P(M|\overline{A} \cap \overline{B})P(\overline{A})P(\overline{B})\\ &= 0.9 \times 0.6 \times 0.8 + 0.8 \times 0.6 \times 0.2 + 0.5 \times 0.4 \times 0.8 + 0.1 \times 0.4 \times 0.2\\ &= 0.432 + 0.096 + 0.16 + 0.008\\ &= 0.696\end{aligned}$$

Question III: What is the probability that contractor is a cowboy by observing building A collapsing?

The question asks for $P(C|A)$, which we do not have any values for by the network. However, by utilising the bayes' rule we can infer:

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

In order to reason about $P(C|A)$, one should first calculate $P(A)$ by the Normalisation Rule. This has been calculated in Question I. Hence:

$$\frac{P(A|C)P(C)}{P(A)} = \frac{0.6 \times 0.1}{0.069} = 0.869$$

Question IV: What is the probability of *M* when *A* is observed?

Since *M* is a child node, we should use the Normalisation Rule for answering this question. The difference here is that $P(A) = 1$, $P(\overline{A}) = 0$ and $P(C)$ should be calculated by the Bayes' rule. This leaves only two possibilities for $P(M|A)$:

$$P(B) = P(B|C)P(C) + P(B|\overline{C})P(\overline{C}) = 0.8 \times 0.869 + 0.02 \times (1 - 0.869) = 0.6978$$

$$P(M) = P(M|A \cap B) \times P(A) \times P(B) + P(M|A \cap \overline{B}) \times P(A) \times P(\overline{B})$$

$$= 0.9 \times 1 \times 0.6978 + 0.8 \times 1 \times (1 - 0.6978) = 0.628 + 0.242 = 0.87$$

2.3 THE SEMANTIC WEB

In this section, we will review the concept of Semantic Web and describe the objectives pursued in this field of research. Section 2.3.1 gives a brief history of the Web. From there we continue to Section 2.3.2 where the aims and overview of the Semantic Web are discussed. Finally, in the latter section, we introduce the tools and technologies from the state of the art.

2.3.1 A BRIEF HISTORY OF THE WEB

In today's modern life style, we take the Web for granted and it plays a major part in our daily lives. However, in the days that computers were slowly being developed and few people knew what a computer actually was, no one could even think of computers as the main infrastructure for future's information exchange. Perhaps, the only person who saw this coming was Vannevar Bush who told the story of a 'photo-electrical' device, making links and moving around documents in a 'microfiche' (Bush, 1945). In 1963, Ted Nelson coined the term 'Hypertext', which is essentially a text with a reference to some resource (Nelson, 1967, Nelson, 1980). Van Dam and Nelson with a number of Brown university students developed a 'Hypertext Editing System' (HES) that could categorise links and branching texts (Carmody et al., 1969). One year later this work was replaced by 'project FRESS', the successor of HES, adding the ability of supporting various terminals (DeRose and Van Dam, 1999). The first practical effort on deploying computers for the so-called 'surfing' happened during 60's when Engelbart and his team made a system called "oNLine System" or NLS (Engelbart and English, 1968) which was capable of manipulating hypertext, email, etc. In 1980, Berners-Lee developed 'Enquire' (Berners-Lee, 1980), a notebook program that allowed links to be made between arbitrary nodes. That was the start for him to come up with the idea of what we now call the Web.

2.3.1.1 WORLD WIDE WEB

In 1989 Berners-Lee wrote "Information Management: A Proposal" (Berners-Lee, 1989) in an effort to convince CERN that a universal hypertext system will fall in the scope of their interest.

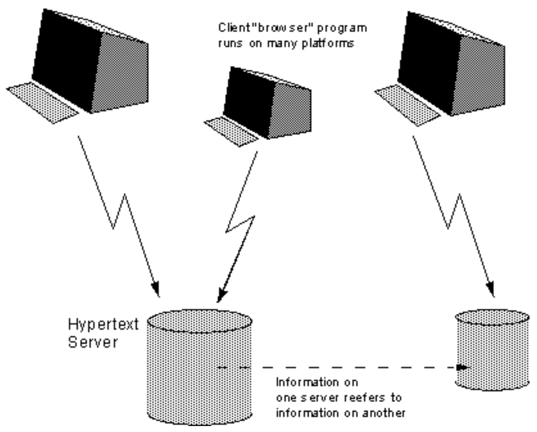

Client "browser" program runs on many platforms
Hypertext Server
Information on one server reefers to information on another

FIGURE 5: PROPOSED CLIENT/SERVER ARCHITECTURE FOR THE FIRST GLOBAL HYPERTEXT SYSTEM AT CERN

Lee started to write the global hypertext system in 1990. He chose the name "World Wide Web" for the project, which contained a GUI browser and editor at that time. 1990-1994 involved exponential growth of the project, in terms of both adoption (1000 fold increase in load of the first web server) and global attention (first and second international WWW conference, both heavily oversubscribed). World Wide Web Consortium (W3C) was founded in mid-94, responsible for organising standards for the newborn web.

2.3.1.2 WEB 2.0

The next major milestone in history of the Web was the introduction of Web 2.0. Darcy DiNucci coined the term in 1999, defining it as a web that "will be understood not as screenfuls of text and graphics but as a transport mechanism, the ether through which interactivity happens" (DiNucci, 1999). In later years, other authors tried to provide a definition for what is meant by Web 2.0, such as (Knorr, 2003). However, the concept of Web 2.0 did not become popular as a paradigm until the first Web 2.0 conference where Tim O'Reilly presented his view over the concept (O'Reilly and Battelle, 2004). He described that by Web 2.0, as the numbering suggests, they mean the "Web as a platform". His example was Google's PageRank algorithm (Brin and Page, 1998) where the way in which a user operates upon the environment affects the search results for the future users. This means that users play an important role in the evolution of the ecosystem, implicitly.

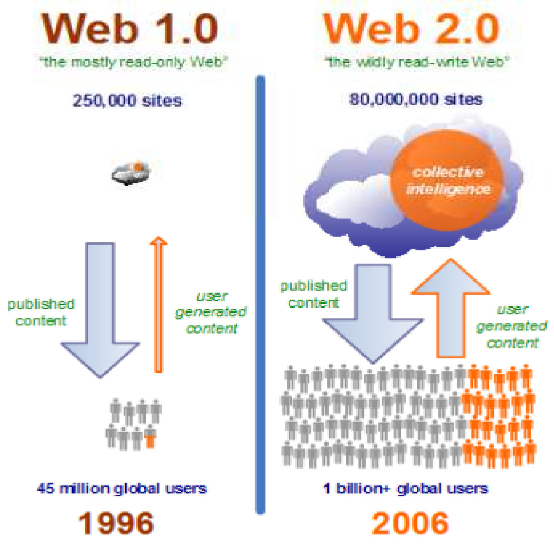


FIGURE 6: WEB 2.0 VS. ORIGINAL WEB,
ADAPTED FROM (THOMAS AND SHETH, 2011)

On the other hand, we have the traditional approach where making any progress is the result of a tightly managed group of experts, hosted by a commercial company, allocating all resources for development of their product. An example could be the variety of commercial encyclopedias the contents of which are provided by the company's experts (employees). Although this method is effective and many of these companies are doing a reasonable job in maintaining the content (i.e. updating the articles before it expires), Wikipedia (a Web 2.0 application) has proven to be just as good, if not better. The difference is specifically substantive when it comes to being 'updated'. There is no such thing as 'update schedule' for articles in Wikipedia applications since users are more than welcome to update the content at their will. Some believe that 'Collective Intelligence' (the Web 2.0 approach) might not be reliable. However, research has provided us with metrics for measuring reliability, such as patterns found by (Thomas and Sheth, 2007) that make it possible to mark a Wikipedia article as 'sufficiently reliable' in a systematic manner.

Web 2.0 refers to a broad range of Web applications, O'Reilly summarises the Web 2.0 principles in (O'Reilly, 2005). A sharp line can be drawn between the applications that could and those that could not be considered as Web 2.0 application. The central factor, as illustrated in Figure 6, is user participation in web modification.

2.3.2 AIMS AND OVERVIEW

Efforts on encoding metadata for the web sites can be traced back to 1997 as a research project called MFC in Apple Computer (Guha, 1997). Later on, W3C decided to adapt similar techniques by introducing the first version of RDF as a web standards (Lassila and Swick, 1999). In 1999, Berners-Lee coined the term 'Semantic Web' and defined it as "a web of data that can be processed directly and indirectly by machines" (Berners-Lee, 1999). Perhaps the best way to understand the enthusiasm and high hopes at the time of emergence is by going back to this famous quote:

> *"I have a dream for the Web become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines." (Berners-Lee, 1999)*

From 1999 onwards, the Semantic Web drew a considerable amount of attention, involving adoption from both academy and industry. Researchers defined what they think the Semantic Web is in two extremes. One group believed that it is the environment that will in turn provide us with the ground infrastructure of human-like intelligence. Others said that it is merely an ecosystem that allows computers to contribute to the process of acquiring better understanding of the data across the web. The W3C has been developing the Semantic Web technologies for more than two decades now and it is the organisation in charge of updating the environment. More precisely, W3C defines 'standards' for the Semantic Web (as well as the original Web), preventing subjective interpretations and unwanted fragmentations. In 2006, Berners-Lee pointed at the notion of 'Linked Data' (Berners-Lee, 2006), stating that useful knowledge should be both meaningful and openly available. He outlined four principles for data, stating that one should:

1. Use URIs for naming things.
2. Use HTTP URIs, allowing users to look up those names.
3. Provide useful information, using the Semantic Web standards.
4. Include links to other URIs, linking data makes it possible to seek extra information about a piece of knowledge.

One can argue that the original aims were optimistic and current state of the Semantic Web does not reflect any of its original goals. In fact, there has been a number of criticisms being imposed against the semantic web , Ivan Herman have summarised the main ones that have been put forward (Herman, 2007):

1. "The Semantic Web is a reincarnation of Artificial Intelligence on the Web"
2. "It relies on giant, centrally controlled ontologies for 'meaning' (as opposed to a democratic, bottom–up control of terms)"
3. "One has to add metadata to all Web pages, convert all relational databases, and XML data to use the Semantic Web"
4. "It is just an ugly application of XML"
5. "One has to learn formal logic, knowledge representation techniques, description logic, etc, to use it"
6. "It is, essentially, an academic project, of no interest for industry"

Below are our comments on each of these criticisms:

1. AI is a heavily investigated field of research, filled with substantially valuable concepts that have been validated scientifically; it is always good to stand on the

shoulder of giants. The central problem of AI has always been 'embodiment' and a worldwide platform, the Semantic Web, is an ideal environment to apply our science and move from theory to practice.

2. For the knowledge to be useful, there should be standards for encoding information. The Web is a perfect example for the achievement of 'organised' approach versus 'scattered' methods. With respect to democracy, the Semantic Web is democratic since one is allowed to put any form knowledge on the Web; it is not the matter of 'content', but the value of 'format'.
3. Technologies from Web 2.0 and ongoing research should help with making these processes automated.
4. Again, it is always good to take a well-established (and in this case widely adopted) approach and extend it.
5. One can contribute to the Semantic Web even by putting CSV data on the web, however, according to Linked Data principles, a piece of knowledge is more useful when has its relations specified. Once again, it is the responsibility of technology to ease the process of conversion. An example could be 'Facebook' where non-technical users are effectively contributing to the knowledge in an intuitive manner. While FOAF vocabulary (Brickley and Miller, 2005) defines a way of formatting knowledge social domains, Facebook is an analogous technology that embodies users in such domain.
6. In section 5.2 we will give an example that will demonstrate the great value of our proposed Semantic Web standard for a sample industry. The same logic holds for all Semantic Web standards.

Although most of the criticisms question the design and/or functionality of the current Semantic Web technologies, there are a limited number of valid 'scientific' issues with the main specifications of the Semantic Web. One issue that falls in the scope of this work is that the Semantic Web, as it is proposed now, is not able to handle uncertainty (Shadbolt et al., 2006, Fensel et al., 2001). Of course, one can define an ontology to express uncertainty within a specific domain, which is exactly the way it is done in previous works described in section 2.4. However, the argument is that such an important feature should be integrated into the knowledge representation language itself. We will discuss this further in section 2.4.2.

2.3.3 TECHNOLOGIES

The Semantic Web is defined as a set of technologies, shown in Figure 7. The key idea is that the Semantic Web is built on top of the well-established standards of the Web. In this section, we briefly review some of the Semantic Web technologies that relate to our work.

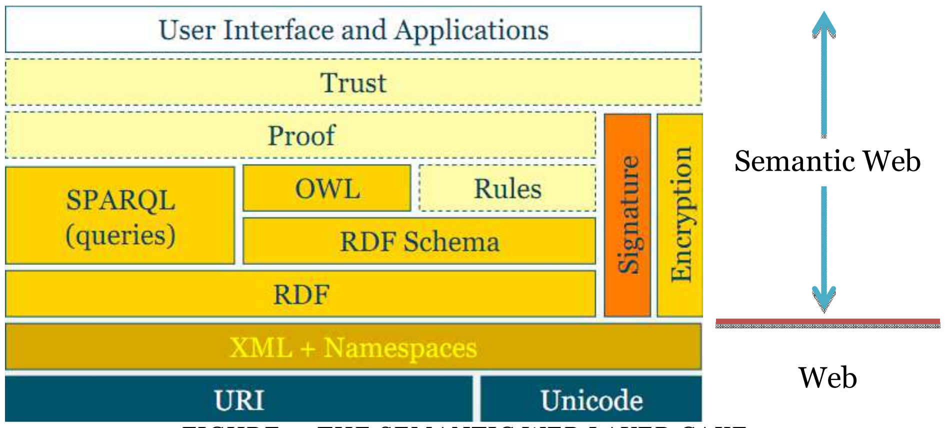


FIGURE 7: THE SEMANTIC WEB LAYER CAKE

2.3.3.1 RDF AND RDF SCHEMA

In broad terms, the Resource Description Framework (RDF) is “a framework for representing information in the Web” (Klyne and Carroll, 2004). In practice however, RDF is the language that allows us to include the description of non-web things in the web. With millions of pages on the Web, there is need for a unified way to elucidate what the content is about in order to make it machine-readable. To fulfill this goal, there should be unified representation formats with rich vocabularies for expressing data about all sorts of content. Current Semantic Web languages utilise URI-based models to describe the semantics in context of the Web, RDF is no exception to that with specific focus on metadata (in fact, other Semantic Web languages, such as OWL, are layered on top of RDF). URIs are standard structures that let us identify every possible entity on and off the Web. Nonetheless, we need more than entities for constructing knowledge, we need to be able to describe facts about different concepts. Furthermore, one should be able to describe relations and dependencies as well as concepts for the knowledge to be useful. That is why most Semantic Web languages follow the Graph data model (which itself is based on Semantic Networks, described in section 2.1)

In the graph data model, we have the notion of triples. A triple describes a fact within a domain. It consists of three fragments, subject, predicate and object; relating the subject to the object by the predicate. In the graph model, this can be illustrated as two nodes being connected by a directed arc:

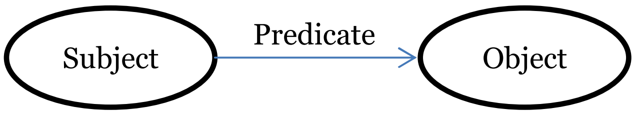


FIGURE 8: A SEMANTIC TRIPLE

A node in RDF can be a URI (optionally with a fragment identifier), a literal or blank. A blank node does not have an intrinsic name, but has a unique reference that differentiates it from other nodes. An RDF graph is a set of RDF triples. To represent different values, XML Schema (xsd) defines ‘Datatypes’, which can be adopted to describe some knowledge in the form of a triple. A datatype can represent any type of data by specifying a lexical space, a value space and a mapping from one to the other. As an example, Table 3 demonstrates how the `xsd:boolean` datatype is defined.

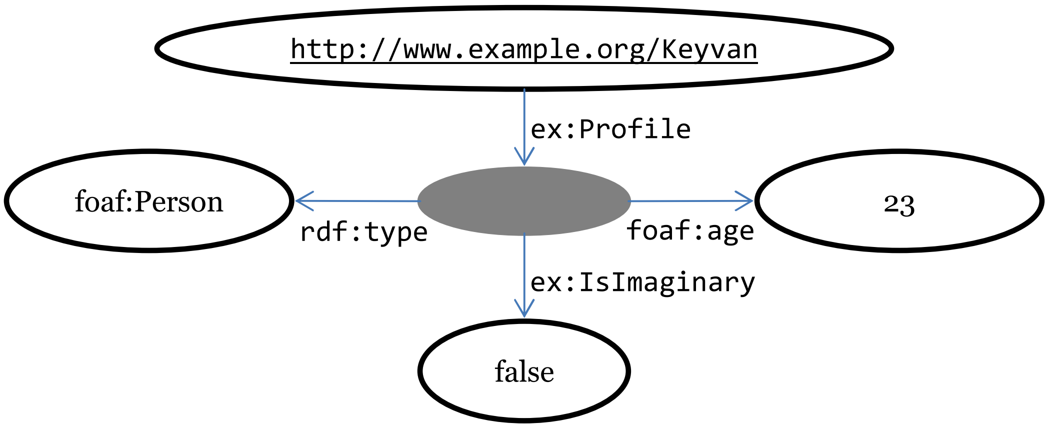


FIGURE 9: USING A BLANK NODE FOR PROVIDING EXTRA INFORMATION

Value Space	{T, F}
Lexical Space	{“0’, “1’, “true”, “false”}
Lexical to Value mapping	{<"true", T>, <"1", T>, <"0", F>, <"false", F>}

TABLE 3: BOOLEAN DATATYPE

RDF Schema (Hayes, 2004) extends RDF, providing further expression and inference capabilities. RDFS vocabulary (Table 4) contains elements for describing relations beyond simple assertions about the world, providing infrastructure for entailment.

`rdfs:Class`	`rdfs:domain`	`rdfs:range`	`rdfs:Resource`
`rdfs:Literal`	`rdfs:subClassOf`	`rdfs:subPropertyOf`	`rdfs:Datatype`
`rdfs:member`	`rdfs:ContainerMembershipProperty`		`rdfs:Container`
`rdfs:seeAlso`	`rdfs:label`	`rdfs:isDefinedBy`	`rdfs:comment`

TABLE 4: RDFS VOCABULARY

Entailment in RDFS is defined in terms of a formal model theory and Semantic conditions. The process of entailment in RDF schema can be characterised by a set of entailment rules shown in Table 5 and Table 6. Entailment rules are to a great degree analogous to principles of Object Oriented paradigm. For instance, reflectivity (defined by **`rdfs10`**, each class is a subclass of itself) or transitivity (**`rdfs11`**). However, entailment in RDFS is not exactly the same as relations in OO languages. One example of aspects that makes them different is that in OO paradigm, predicates can only connect two classes (concepts) whereas in RDF, a triple can have a subject that is used as a predicate in another triple. Thereby, we do not have such thing as transitive properties in OO languages like there is in RDF schema (**`rdfs5`**).

`se1`
`s p o =>`
`s p _:bNode`
	Where _:bNode *identifies a blank node allocated to* o *by rule* **`se1`** *or* **`se2`**.	**`lg`**
		`v1 v2 LITERAL =>`
		`v1 v2 _:bNode`
			Where _:bNode *identifies a blank node allocated to the literal by this rule.*

`se2`
`s p o =>`
`_:bNode p o`
	Where _:bNode *identifies a blank node allocated to* s *by rule* **`se1`** *or* **`se2`**.	**`gl`**
		`_:bNode v1 v3 =>`
		`v1 v2 v3`
			Where _:bNode *identifies a blank node allocated to the literal by rule* **`lg`**.

TABLE 5: SIMPLE ENTAILMENT RULES (SE),
LITERAL GENERALIZATION (LG) AND LITERAL INSTIGATION (GL)

`rdfs2`
`{ s1 rdfs:domain o1 .`
`s2 s1 o2 . } =>`
`s2 rdf:type o1.`
	`rdfs3`
	`{ s1 rdfs:range o1 .`
	`s2 s1 o2 . } =>`
	`o2 rdf:type o1 .`

`s rdf:type rdfs:Class . =>`
`{ s rdfs:subClassOf rdfs:Resource` **`rdfs8`**
`s rdfs:subClassOf s .` **`rdfs10`** `}`
	`rdfs6`
	`s rdf:type rdf:Property . =>`
	`s rdfs:subPropertyOf s.`

`rdfs11`
`{ s1 rdfs:subClassOf o1 .`
`o1 rdfs:subClassOf o2 . } =>`
`s1 rdfs:subClassOf o2 .`
	`rdfs9`
	`{ s1 rdfs:subClassOf o1 .`
	`s2 rdf:type s1 . } =>`
	`s2 rdf:type o1 .`

`rdfs5`
`{ s1 rdfs:subPropertyOf o1.`
`o1 rdfs:subPropertyOf o2 . } =>`
`s1 rdfs:subPropertyOf o2 .`
	`rdfs7`
	`{ s1 rdfs:subPropertyOf o1 .`
	`s2 s1 o2 . } =>`
	`s2 o1 o2 .`

`s1 p1 o1 =>`
`{ s1 rdf:type rdfs:Resource .` **`rdfs4a`**
`o1 rdf:type rdfs:Resource .` **`rdfs4b`**
`p1 rdf:type rdf:Property .` **`rdf1`** `}`
	`s p LITERAL. =>`
	`{ _:bNode rdf:type rdfs:Literal .` **`rdfs1`**
	`_:bNode rdf:type rdf:XMLLiteral .` **`rdf2`** `}`

`rdfs12`
`s rdf:type`
`rdfs:ContainerMembershipProperty . =>`
`s rdfs:subPropertyOf rdfs:member .`
	`rdfs13`
	`s rdf:type rdfs:Datatype . =>`
	`s rdfs:subClassOf rdfs:Literal .`

TABLE 6: RDF AND RDFS ENTAILMENT RULES

One should note that the entailment rules are ‘informative’, not normative. They have been designed to characterise the ‘behaviour of a reasoner’, not the process of entailment itself. Figure 21 shows the example of an entailed triple based on the described rules.

```
<ex:worksFor> <rdfs:domain> <foaf:Person> .
<ex:worksFor> <rdfs:range> <ex:Company> .
<ex:Peter> <ex:worksFor> <ex:Microsoft> .
```

```
SPARQL:
SELECT ?company
WHERE
 { ?x a ex:Company . }
```

company
Microsoft

FIGURE 10: RDFS USAGE EXAMPLE

2.3.3.2 OWL

The Web Ontology Language (OWL) is the W3C standard for designing ontologies on the web. It provides necessary elements for representing knowledge within a domain by describing concepts and relations (Smith et al., 2004). Moreover, OWL will allow one to:

1. Formalise a domain by defining classes and properties of those classes.
2. Define individuals and assert properties about them.
3. Reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

In 2000, DARPA introduced DAML in an effort to make the content on the web more understandable for machines (Hendler and McGuinness, 2000). Later in 2001, the Ontology Inference Layer (OIL) has been developed (Fensel et al., 2001) based on frame logic and the *Joint EU/US Committee on Agent Markup Languages* decided to combine DAML and OIL according to (Lacy, 2005). DAML+OIL (Horrocks, 2002) designed to integrate semantic formalism from Description Logic. Followed by W3C replacing the DAML+OIL project with OWL by introduction of OWL 1.0 in 2002.

The underlying logic behind the web ontology language is Description Logic. Therefore, the level of expressiveness for an ontology described by OWL can be denoted by DL conventions described in section 2.1.2. In addition, adopting certain vocabularies from OWL can change the level of expressiveness. For example, adding `owl:hasValue` (object value restriction) and `owl:cardinality` (numbering) elements to an ontology that is currently of $\mathcal{ALC}$ expressivity will result in going to $\mathcal{SHOIN}$ level. As mentioned in section 2.1.2, there is a trade-off between expressivity and the speed of reasoning. This is one of the design aspects that need to be considered for any ontology encoded by OWL since the reasoning should be done in a time for the knowledge to be useful.

OWL 2 (Motik et al., 2009) is the successor of the original OWL, adding the following features to the Web ontology language:

- Constructs for increased expressivity
- Datatype support
- Metamodelling
- Annotation

2.3.3.3 N-TRIPLES

N-Triples has been proposed as test-case format by RDF Core working group, defined as a line-based, plain text format for encoding an RDF graph (Grant and Beckett, 2001). The simple syntax of N-Triples makes it an ideal candidate for the purpose of testing. In this work, we have adopted the syntax of N-Triples to demonstrate that a syntax can be more powerful than an ontology with aims of fitting uncertainty into the Semantic Web.

Given a subject, a predicate and an object, a triple can be encoded by N-Triples as follows:

<Subject> <Predicate> <Object> .

2.4 SEMANTIC WEB MEETS UNCERTAINTY

In 2005, Sheth and his colleagues pointed at the notion of 'Soft Semantics', arguing that the Semantic Web currently puts too much of emphasis on pure formal logic (Sheth et al., 2005). In other pieces of research, arguments can be found which state that probabilities should be taken into account in the process of reasoning in the future of the Semantic Web (Fensel and Van Harmelen, 2007, Shadbolt et al., 2006). In this section, we review the works relating to uncertainty in the Semantic Web. Validation of the knowledge is an important issue when the knowledge is acquired at the scale of the Web. This is where we can identify an overlap between the Semantic Web and uncertainty; researchers are working on this problem in a subfield of the Semantic Web called Trust. On the other hand, there are situations in which some piece of knowledge is not certain and the uncertainty has not been imposed by the knowledge source, but by the nature of the knowledge. Uncertainty has been heavily investigated in traditional knowledge management techniques, we have reviewed some relating works in section 0. In this section, we give a brief review over the issue of Trust in the Semantic Web, then we continue with describing works done in a new area of research on uncertainty, integrating Bayesian Networks with the Semantic Web.

2.4.1 TRUST IN THE SEMANTIC WEB

Where to acquire knowledge from and trustworthiness of the sources has always been a controversial topic in research on the Semantic Web. Keeping the web an open place that everyone can contribute to is the basic concept that makes it the best environment for accumulating knowledge. However, the very same concept can become detrimental since the consistency of the knowledge can be impaired by inaccurate data. The openness of the semantic web has been defined by the 'Open world assumption' rule, as described in OWL's definition "OWL makes an open world assumption. That is, descriptions of resources are not confined to a single file or scope. While class C1 may be defined originally in ontology O1, it can be extended in other ontologies" (Smith et al., 2004). The same rule has been adapted in RDF, "RDF is an open-world framework that allows anyone to make statements about any resource. In general, it is not assumed that complete information about any resource is available"(Klyne and Carroll, 2004). Now the question is how to manage the reliability of the sources in a way that does not inflict any restrictions over the web. One approach to Trust handling is by providing metrics for measurement of trustworthiness, such as a work by (Golbeck et al., 2003). The other approach is by utilising the concept of probabilities, like in (Richardson et al., 2003). A future direction for this field could be defining the trust relations by Bayesian Networks. This work can be the basis of such extensions for Trust in the Semantic Web.

2.4.2 BAYESIAN NETWORKS AND THE SEMANTIC WEB

In section 2.2.2 we reviewed the concept of Bayesian Networks and provided an example that showed the powerful reasoning abilities of the concept. As described in section 1, the motivation behind this dissertation is to extend the Semantic Web with the expressivity and reasoning abilities of the Bayesian Networks'. We have identified two pieces of work done by others with the same aims we pursue in this dissertation (Costa et al., 2008, Ding and Peng, 2004). Despite the different titles, these works have followed the same approach in their design. Moreover, they propose extensions for OWL, each providing an ontology enriched with vocabulary that can describe the probabilistic relation in a domain based on the Bayesian Networks theory.

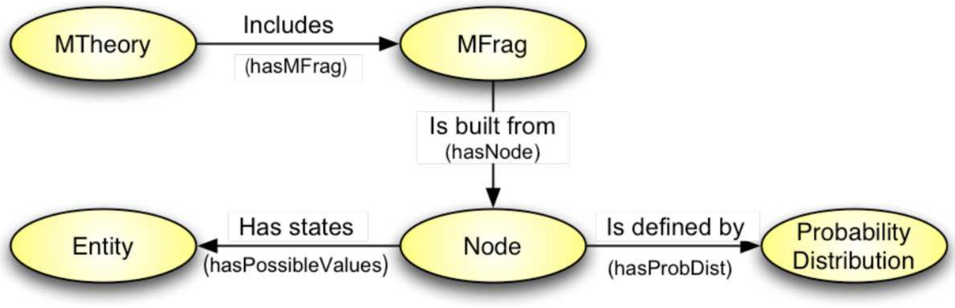


FIGURE 11: PR-OWL ONTOLOGY, ADAPTED FROM (COSTA ET AL., 2008)

The question here is why these works fail to address the research question described in section 1? Furthermore, we claim that any extension for the Sematic Web in form of an ontology will not be the best way to integrate Bayesian Networks with the Semantic Web. We argue that ontology level extensions are not robust enough to be adopted in a worldwide scale. In contrast, this work does not have these limitations since it adopts the concepts of quads instead of triples provided by traditional semantic web languages such as OWL. Not only does this method enables us to establish the uncertainty within the interconnections of a semantic node, but also it reveals us from having to add extra dependencies from some other ontology to every single triple in our specific application. Adding these extra dependencies to ontology makes it less understandable and inflicts overhead when it comes to reasoning. From a user's point of view, having to deal with an unfamiliar ontology for consuming the knowledge from another domain is not reasonable. In section 4.1, we will describe why defining an ontology for representing uncertainty is not the best way for integrating Bayesian Networks with the Semantic Web and present our alternative solution.

3 REQUIREMENT ANALYSIS

The goal that we pursue in this work is to demonstrate that a shift in our view over the integration of uncertainty in the Semantic Web could have a dramatic influence on both effectiveness and usability of knowledge management. The aim is to understand how one can fit uncertainty into the Semantic Web. In addition, it should be taken into account that we are not building a scale, instead, we are providing a proof of concept. The field of computer science has already experienced situations where a slight shift in methodology has a substantial impact on the usability of the system as a whole, although it might have looked simple at the beginning. Perhaps the best example is the WWW itself; there were efforts on deploying computers for automating the process of information exchange, but it only became possible when Berners-Lee gave a comprehensive rethought to the way we used to think about information (i.e. having a web of data instead of isolated data centres scattered across the globe). In this section, we will go through requirements for building a system that enables us to evaluate the applicability of the proposal.

3.1 USER SCENARIOS

We need to have a clear description of the scenarios within which users interact with our system to fulfill their objectives. In this section, we will give examples of scenarios within which users can adopt our system to achieve their goals.

3.1.1 SCENARIO I, CRISP REASONING

Fred wants to buy a camera with specific features. He has access to a database that contains knowledge about different models and their specifications, encoded by RDF. Fred has a clear description of what he wants to buy, a digital camera with a resolution of above 12 megapixels.

3.1.2 SCENARIO II, REPRESENTING BELIEF

Arthur is an accountant. On his computer, he has a large amount of data about different products that a company sells. He wants to publish this information for others to use. However, he is not certain about the validity of the information that he has. In addition, the database is so large that it cannot possibly be processed by hand and therefore, if others require to use this information, they should be able to do it in a systematic way. Arthur wishes he could put the data online and describe how much he believed the information is correct.

3.1.3 SCENARIO III, BAYESIAN REASONING

Mr. Richardson is a business owner at the edge of bankruptcy who needs an effective solution as a matter of urgency. He describes the business as tourism, where they sell stuff to people who come to enjoy the sunny weather in a beach. From his years of experience, Mr. Richardson can give us a detailed description of the relations in the market that they work in. Moreover, he knows that when the sea level is low and it is less humid therefore, weather is more likely to be sunny. In addition, travelers can enjoy the beach since it is not covered by water. Richardson suggests that if it were possible to predict the sea level within the margin of a day, they would have a dramatic increase in profit. The task here is to investigate the possibility of adopting N-Quads for modeling this domain as a Bayesian Network and evaluate the usability of such solution.

3.2 FUNCTIONAL REQUIREMENTS

By having our requirements specified, we will be able assess how useful and effective an implementation is by conducting qualitative evaluation. In order to prove the hypothesis, a number of requirements are illustrated below.

The system should provide the user with tools for:

1. Encoding knowledge by a knowledge representation language: Involves designing a language, i.e. syntax and specifications, and a development environment for deploying the language.
2. Representing uncertainty within the language: This is the key element that distinguishes this work from previous efforts. Uncertainty should be represented based on Bayesian networks' principles.
3. Parsing data from a file: User stores the written code in form of a file and a parser should be able of extracting the knowledge from a given file.
4. Visualising knowledge: A user interface, capable of showing the parsed data in a form that is analogous with a Bayesian network.
5. Reasoning over the knowledge: A reasoner that implements reasoning in Bayesian networks, with the ability to execute queries. Here is an example of what is meant by a query: *"The probability of U given V"*
6. Interaction between user and the system: A graphical user interface that hosts all of the above-mentioned capabilities.

The first two entries in the list above are design requirements whereas the remainder should be satisfied by the implementation. The design of the language needs to be well specified and guaranties that all the implementation that follow the design principles will produce the same behaviour.

3.3 NONE-FUNCTIONAL REQUIREMENTS

Since we are not building a scale, none-functional requirements such as optimised implementation will not be relevant. As we are only building a proof of concept, these requirements are secondary to the functional requirements described in the previous section. We have identified two non-functional requirements that fall in the scope of this work:

1. Maintainability: Although the code does not have to be optimised, it should follow the design principles of a maintainable system. Furthermore, the implementation should be in the form of a software framework that allows maintainability and optimisation.
2. Readability of the syntax: The syntax of the knowledge representation language should be understandable.

4 DESIGN AND IMPLEMENTATION

In this section, we review the design and implementation of the system that we developed for addressing the requirements described in the previous section. Designing a new language is certainly not an easy task and includes both low-level and high-level aspects. Moreover, low-level tasks involve reading text character by character or forming a binary truth table, and high-level tasks like extracting the semantics from a given input. In a normal software project, it is usually possible to re-use the work already done by others and build a system on top of them. However, this had not been the case given the requirements of this work and all of the components were designed and developed from the ground up.

4.1 N-QUADS: AN EXTENSION TO N-TRIPLES

As of now, there is not any Semantic Web language equipped with probabilistic inference capabilities. In this dissertation we present 'N-Quads', an extension for the Semantic Web with the ability of handling uncertain knowledge. N-Quads inherits the power of probabilistic reasoning from Bayesian Networks and simplicity of the syntax from N-Triples. The reason behind choosing the name of 'Quad' is that a Quad is essentially a triple plus uncertainty.

In section 2.4.2, we described two previous works on the integration of Bayesian Networks and the Semantic Web. The difference between our proposal and the previous ones is that our design captures reasoning abilities of Bayesian Networks in form of a language whereas previous designs have done it as ontologies. There are several reasons on why uncertainty handling should move from ontology level to language level:

1. Uncertainty is ubiquitous, even the facts that have been affirmed as purely accurate can turn uncertain as a result of new discoveries.
2. Triples are only capable of relating three URIs (namely, subject, predicate and object) and an ontology can only use triples for specifying the relation from one URI to another. This nature of triples does not allow the addition of uncertainty within a single triple.
3. User adoption in a worldwide scale is the only way for the probabilistic relations from different domains to be useful in the form of knowledge. It is commonsensical that only simpler designs can gain popularity in the masses web structures. We argue that the design of a language will be much simpler than an ontology
4. For one to use a Bayesian ontology, one should first learn the design of the ontology and then specify probabilities for every single triple in the domain by using a new set of triples that relate the knowledge which is already in place to the ontology. Even for Semantic Web experts, that is a hard task to accomplish.
5. Having dependencies on extra ontologies only means longer reasoning time. We claim the process of reasoning in our design will be more robust with less overhead, since uncertainty has been specified in each assertion instead of extra dependencies.
6. By defining a new language, we would no longer be limited to the restrictions of triples. In such a language, uncertainty can lay in the place that it belongs to, right after the triple, making it intuitive and easy to understand and reason with.

Syntax plays a major role in the scope of this work since it can be perplexing. On one hand, it should be expressive enough to satisfy the requirements of section 2.4.1. On the other, it needs to be simple enough to demonstrate how compact and unambiguous the implementation could be. In addition, it should be designed in a way to be familiar for the Semantic Web community, Bayesian Theory researchers and people with general computing background. Being fluently readable is a general requirement that should be taken into account for any type of syntax design, whether it is for a programming language or a Semantic Web technology.

We designed N-Quads to be backward compatible, and by this we mean a triple can be considered as a Quad without any modification. A quad as such is named as a 'crisp' quad, which means the probability of the knowledge represented by the triple is 'one'. Accordingly, we have the minimal configuration of a Quad as follows:

<Subject> <Predicate> <Object> .

Where Subject, Predicate and Object each represent a URI.

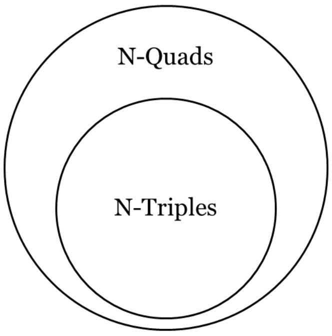


FIGURE 12: N-QUADS BUILDS ON TOP OF N-TRIPLES

Another configuration is by having a single number in the list of uncertainty values. This means that the extent to which we are certain about a piece of knowledge is given by a probability. The resulting Quad is called a 'Belief' Quad:

<Subject> <Predicate> <Object> { P } .

Where P is the probability and $0 \leq P \leq 1$

A 'relational' Quad is one that does not represent any facts and instead, describes a relation between some pieces of knowledge in the domain. Such Quad is formed from three fragments, a 'Proposition', 'Uncertainty Values' and a list of 'Givens'.

$$
\begin{array}{l}
<S_p> <P_p> <O_p> \{ P_1, \dots, P_N \} \\
<S_{G_1}> <P_{G_1}> <O_{G_1}> \\
\dots \\
<S_{G_M}> <P_{G_M}> <O_{G_M}> .
\end{array}
$$

Where $2^M = N$

```
<ex:Contractor> <ex:is> <ex:Cowboy> {0.1} .
<ex:BuildingA> <ex:status> <ex:Collapsed>
{0.01,0.6}
       <ex:Contractor> <ex:is> <ex:Cowboy> .
<ex:BuildingB> <ex:status> <ex:Collapsed>
{0.02,0.8}
       <ex:Contractor> <ex:is> <ex:Cowboy> .
<ex:Microsoft> <ex:status> <ex:Bankrupt>
{0.1.0.5,0.8,0.9}
       <ex:BuildingA> <ex:status> <ex:Collapsed>
       <ex:BuildingB> <ex:status> <ex:Collapsed> .
```

FIGURE 13: MICROSOFT'S BUILDINGS EXAMPLE IN N-QUADS

4.2 ARCHITECTURE

N-Quads files are stored in the memory with '.N4' extension. We need to design a system that can interpret an N4 file and extracts the knowledge encoded by the syntax. Such system should be able to:

1. Parse the syntax, given an N4 file.
2. Generate objects that encapsulate the knowledge extracted from the given file.
3. Reason about the knowledge, whether it is crisp or relational.

According to the conditions above, we designed the architecture shown in Figure 14. We need to design two modules, a *Parser* and a *Reasoner*, and implement the required sub-components of each. We will describe how this is done in next sections.

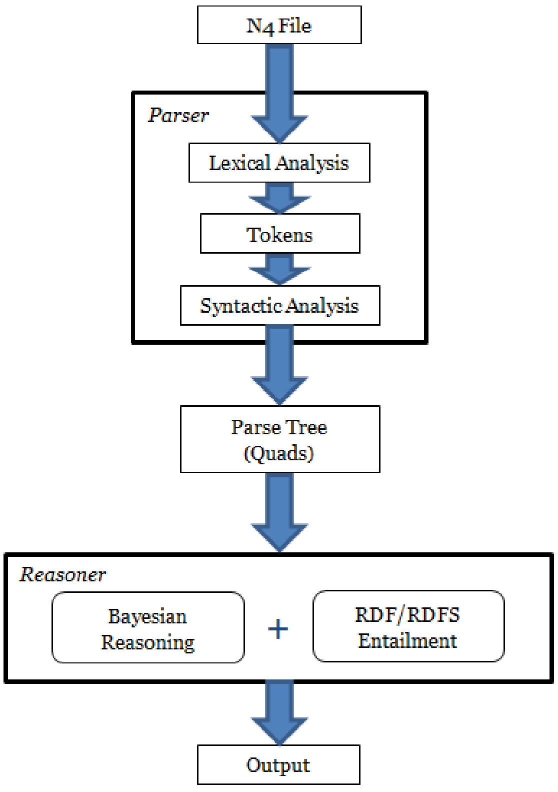


FIGURE 14: DESIGN ARCHITECTURE

4.3 PARSER

The first layer of the architecture presented in Figure 14 is the *Parser*, which is responsible for generating Quads (parse tree) from the given input. The way in which we implemented the parser is by identifying the tokens based on the N-Quads' syntax and conducting the lexical and syntactical analysis in parallel by utilising the concept of state machines. Our parser is essentially formed from three Deterministic Finite Automatons (Rabin and Scott, 1959, McCulloch and Pitts, 1943) illustrated in Figure 15, Figure 16 and Figure 17. The first DFA is designed to identify a valid triple. It does so by identifying the URIs that form a triple, one by one. The three main states of this state machine are 'S', 'P' and 'O', representing 'Subject', 'Predicate' and 'Object', respectively.

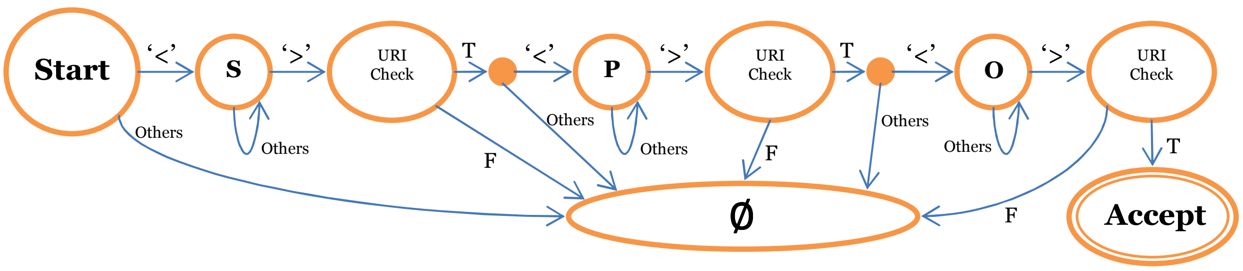


FIGURE 15: DFA1, TRIPLE IDENTIFIER

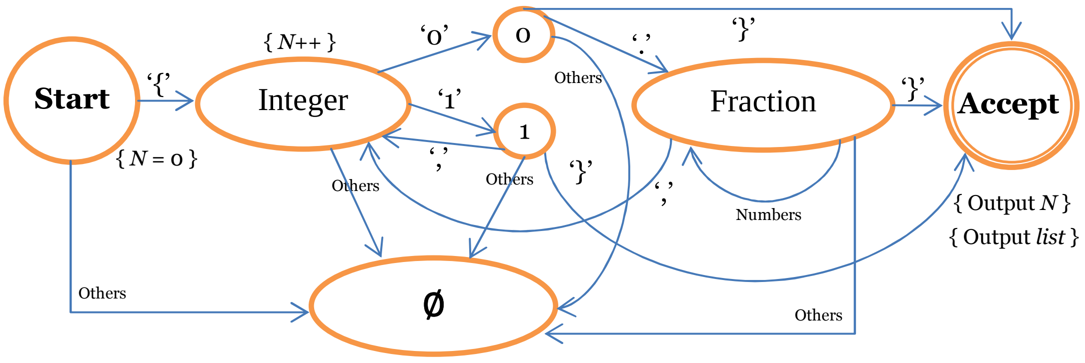


FIGURE 16: DFA2, UNCERTAINTY VALUES

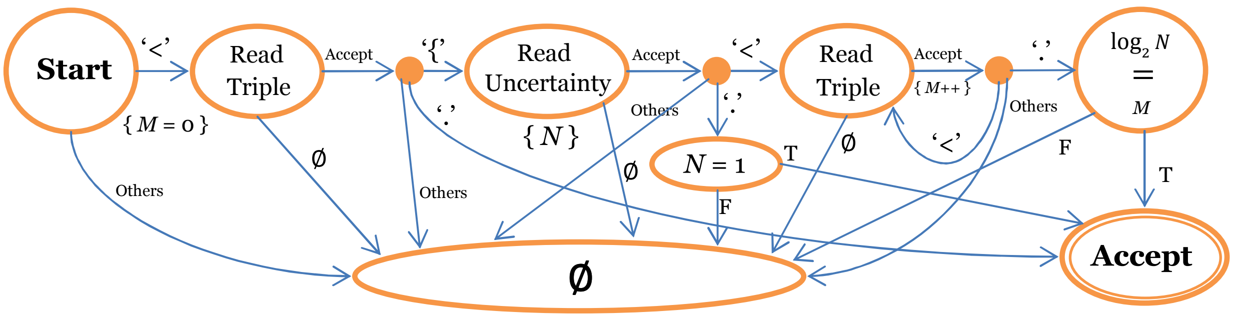


FIGURE 17: DFA3, QUAD IDENTIFIER (I.E. SYNTACTICAL ANALYSIS)

According to the design of our syntax, a URI in N-Quads is surrounded by a '<' and a '>' character. Consequently, we can have a transition from state 'Start' to state 'S' by observing the '<' character in DFA1. When arrived at state 'S', the parser keeps reading the input stream until it reaches the '>' character, which notifies that the subject URI has been read completely. From there, we have a transition to the 'Check URI' state, that is, a validation procedure to check if the lexeme read in the previous state is applicable. This is done by calling the constructor method of the 'Uri' class and passing the lexeme as the input argument. In case that the provided lexeme is applicable, an instance of the URI will be allocated to the corresponding token. Otherwise, the constructor method will throw an exception, stating that the provided string (the lexeme) is not a valid URI. Accordingly, there are two possible transitions from 'Uri Check'; one is when the lexeme could be translated to a URI (denoted by the arc labeled as T), and the other is when it is not valid (denoted as F). In case that any inconsistency is detected, we will have a transition to the 'Trap State' (∅). This indicates that the given input has not been qualified in Lexical Analysis phase. For example, if we had '$' as the first character of the stream, we would have a transition from 'Start' to '∅'. The reason is that there are only two outgoing arcs from 'Start', namely, '<' and 'Others'. Since '$' is not equal to '<', it will be considered as 'Others' by the parser and the transition by the corresponding arc will lead us to the trap state, resulting in a 'parse exception'. Following the same logic, three URIs will be read from the input and if there are no errors, the triple will be accepted and the corresponding tokens will be passed through the next stage. Examples of a valid and an invalid triple are shown in Table 7.

<ex:Human> <ex:Has> <ex:Mind>

Lexeme	Token Type
<	URI Start
ex:Human	URI (valid)
>	URI End
<	URI Start
ex:Has	URI (valid)
>	URI Start
<	URI End
ex:Mind	URI (valid)
>	URI End

<ex:People> <ex:Have> <$Dreams$

Lexeme	Token Type
<	URI Start
ex:People	URI (valid)
>	URI End
<	URI Start
ex:Have	URI (valid)
>	URI End
<	URI Start
$Dreams$	*Invalid URI*

TABLE 7: EXAMPLE OF VALID AND INVALID TRIPLES

Next in the lexical analysis, we have the 'Uncertainty Values'. As described in Section 4.1, these are values corresponding to the 'given' fragment of a Quad. According to our syntax principles described in section 4.1, uncertainty values are surrounded in braces, separated by commas and should be in range of zero to one. In order to adapt this principle, DFA2 has been implemented in the parser. The first state in DFA2 starts by reading '{' from the input, which indicates that there should be at least one probability value in the list to come. Therefore, in case that anything other than a number is to be read next from the input, the token should not be accepted. In addition, the numbers should fall in the specified range by the syntax. Accordingly, we have a state called 'Integer' in the DFA that only accepts values of 0 and 1. If the number starts by 0, fractions are allowed by transition with '.' to state 'Fraction'. In contrast, observing '1' means that the value should not have any fractions, since the values greater than one are not allowed. The other options after '1' is to continue with a ',' for the remainder of the list or '}' for finishing it. There is also the notion of N, the number of processed values so far. Each time that we arrive at state 'Integer', we should increase the value of N by one. We do so because the value of N will be needed later on for syntactical analysis. A closed brace indicates the end of the list and moves us to the final state where the list and the value of N should be passed as outputs.

DFA3 is the core unit of the parser, responsible for conducting Syntactical Analysis. It utilises the previously mentioned DFAs to achieve this purpose, allowing the lexical analysis to be performed in parallel. Table 8 shows a complete trace for a valid sample. The goal of DFA3 is to identify applicable quads in order to form the parse tree for the next stage. This has been made possible by calls to DFA1 and DFA2, which in the diagram are denoted as 'Read Triple' and 'Read Uncertainty', respectively. The DFA starts by reading a single triple using DFA1, and then continues to uncertainty by processing the probability values (DFA2). From there, it moves on to identifying the given triples by recursive use of DFA1 until it reaches character '.', which identifies the end of the statement. Finally, the validity of the uncertainty values in the list will be tested against the number of given triples by comparing the values of N and M. In a valid Quad, the number of uncertainty values is exponential in the number of given facts with a base of two. Accordingly, we can identify an applicable Quad by checking if the equation of '$\log_2 N = M$' holds true. When a valid Quad is identified, it will be added to the parse tree. An instance of the class 'ParseTree' stores the quad and connects the corresponding dependencies that the Quad suggests.

```
<ex:Person> <ex:is> <ex:Happy> {0.2, 0.7}
<ex:World> <ex:is> <ex:Beautiful> .
```

Symbol	DFA	State	Output
	3	Start {M=0}	-
‘<’	1	S	-
“ex:Person”	1	S	-
‘>’	1	URI Check	-
T	1	●	-
‘<’	1	P	-
“ex:is”	1	P	-
“>”	1	URI Check	-
T	1	●	-
‘<’	1	O	-
“ex:Happy”	1	O	-
‘>’	1	URI Check	-
T	1	Accept	-
Accept	3	●	-
‘{’	2	Integer {N=1}	-
‘0’	2	0	-
‘.’	2	Fraction	-
‘2’	2	Fraction	-
‘,’	2	Integer {N=2}	-
‘0’	2	0	-
‘.’	2	Fraction	-
‘8’	2	Fraction	-
‘}’	2	Accept	N = 2, list = {0.2, 0.8}
Accept	3	●	-
‘<’	1	S	-
“ex:World”	1	S	-
‘>’	1	URI Check	-
T	1	●	-
‘<’	1	P	-
“ex:is”	1	P	-
“>”	1	URI Check	-
T	1	●	-
‘<’	1	O	-
“ex:Beautiful”	1	O	-
‘>’	1	URI Check	-
T	1	Accept {M=1}	-
Accept	3	●	-
‘.’	3	$\log_2 N = M$	-
T	3	Accept	-

TABLE 8: SYNTACTICAL AND LEXICAL ANALYSIS FOR A SAMPLE CODE

The approach that we took in the code implementation was to generate the framework objects on-the-fly. Moreover, parsing the data and initiation of the components take place in parallel and instead of having abstract tokens and mapping from them to real objects, the tokens are defined as the objects themselves. The bottom class in the hierarchy is ‘Uri’, a simple data structure that encapsulates a URI, providing tools such as extracting the scheme of the URI, access to different fragments and most importantly, instantiating an object from a given string. The constructor method of the Uri class can be adapted to act as the “Check URI” state in DFA1. ‘SemanticTriple’ is the next component, composed of three instances from the Uri class. An operation named “Add” has been provided, allowing URIs to be added while the process of parsing is still ongoing. In addition, the static method ‘FromSPO’ makes it possible to generate triples when all the three URIs are present. This feature will be used in the reasoning where an entailment rule might have all of its elements specified within itself.

A simple hierarchy of Quad elements has been designed that allows tokens of types ‘UriElement’ and ‘UncertaintyElement’ to be added to the quad dynamically. The transition from the ‘Start’ state with ‘<’ in DFA1 will be followed by creating an instance of ‘UriElement’ that eventually will be added to a semantic triple as the ‘Subject’. Two more URIs are read from DFA1, filling the ‘Predicate’ and ‘Object’ slots of the triple, respectively. At this point, we have a valid triple, which we then set as the proposition of the Quad. The next element of a Quad is the list of uncertainty values. The token that we generate after reading this list by DFA2 is an instance of ‘UncertaintyElement’. The final fragment of a Quad is its list of givens. DFA3 notifies when a triple has been read in the given section. We respond that by adding the triple to the list of givens of our Quad. This cycle continues until the character ‘.’ is read from the input stream that indicates the end of the statement. By reaching this point, we need to validate the list of uncertainty values obtained in the previous steps. We have already described how this is done in DFA3. When the validity of the Quad is confirmed, we raise the corresponding flag, ‘IsValid’. The process of parsing the Quad is completed and the corresponding object will be added to our parse tree.

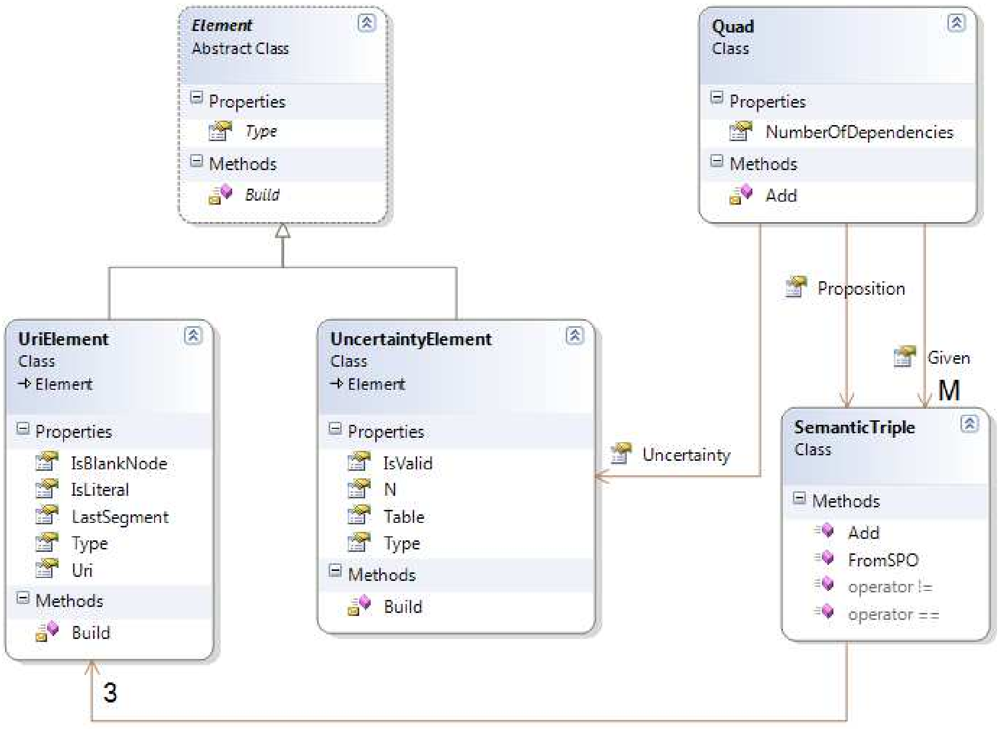


FIGURE 18: DESIGN OF A QUAD

In general, a reasoner is responsible for drawing conclusion from a set of given facts. In the scope of this work, the reasoner should be equipped with a mechanism of traditional reasoning as well as probabilistic inference. Accordingly, our reasoner consists of two main components, an 'Expert System' engine and a 'Bayesian Reasoner'. The first component will provide us with the ability to reason over crisp knowledge. A module has been implemented to provide the system with this capability, called 'SemanticEntailmentAgent'. An instance of this class can be made by passing the Quads generated by the parser to the constructor method of the class. Since the knowledge is crisp, this module only evaluates the 'Proposition' fragment of the quad, testing it against the rules stored in the knowledge base. The rules corresponding to this module are hard-coded by providing each rule's specifications. An abstract class named 'Rule' is designed for encoding the specifications of a rule.

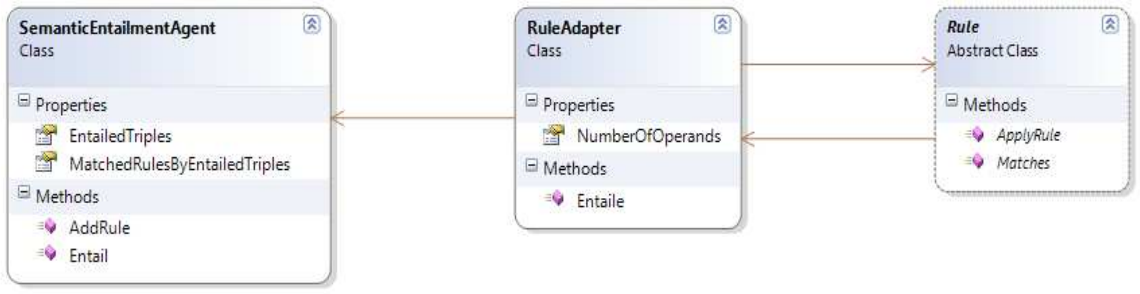


FIGURE 19: FRAMEWORK COMPONENTS FOR CRISP REASONING

One can define a rule by creating a sub class of the 'Rule' class and overriding two simple operations of this class, namely, 'Matches' and 'ApplyRule', which can translate to the 'IF' and 'THEN' parts of the traditional rule format. In addition, one should specify how many premises the rule has by supplying the number of premises (number of 'Operands' in our terminology) to the constructor of the base class. For instance, assume that 'John gets wet' when 'Weather is rainy', 'John does not have an umbrella' and 'John is outside'; Figure 20 shows how this rule can be defined using the framework's hook methods.

```
class Rule1 : Rule
{
    public Rule1() : base(3) { }

    public override bool Matches(List<SemanticTriple> triples)
    {
        if (triples[0].Subject.Uri == new Uri("ex:John")
            && triples[0].Predicate.Uri == new Uri("ex:is")
            && triples[0].Object.Uri == new Uri("ex:Outside"))
                if (triples[1].Subject.Uri == new Uri("ex:John")
                && triples[1].Predicate.Uri == new Uri("ex:doesNotHave")
                && triples[1].Object.Uri == new Uri("ex:Umbrella"))
                    if (triples[2].Subject.Uri == new Uri("ex:Weather")
                    && triples[2].Predicate.Uri == new Uri("ex:is")
                    && triples[2].Object.Uri == new Uri("ex:Rainy"))
                    return true;
        return false;
    }

    public override List<SemanticTriple> ApplyRule(List<SemanticTriple> triples)
    {
        return new List<SemanticTriple>()
        {
            SemanticTriple.FromSPO(new Uri("ex:John"), new Uri("ex:is"), new Uri("ex:Wet"))
        };
    }
}
```
FIGURE 20: DEFINING A CRISP RULE USING THE THE FRAMEWORK

The basic logic behind our crisp reasoning functionality is analogous to that of expert systems. We have a triple base, called 'Agenda', which is initially filled with the extracted triples from the parsing phase. Each instance of the class 'Rule' has a 'RuleAdapter' associated with it, which checks the rule against all of the triples in the agenda and applies the rule if the conditions are satisfied. The entailment agent supplies all of the rule adapters with the current triples in the agenda and in the case that a new triple is entailed by a rule, it will be added to the agenda (if it does not already exist). It is also possible for a rule to have more than one conclusion; in this case, all of the entailed triples will be added. Once all of the rules have been checked and applied (if appropriate), the reasoning agent will compare the number of triples in the agenda before and after the process. If it identifies that one or more triples have been added to the agenda, it repeats the process all over again. The agent will confirm the end of reasoning when the number of triples in the agenda stops increasing.

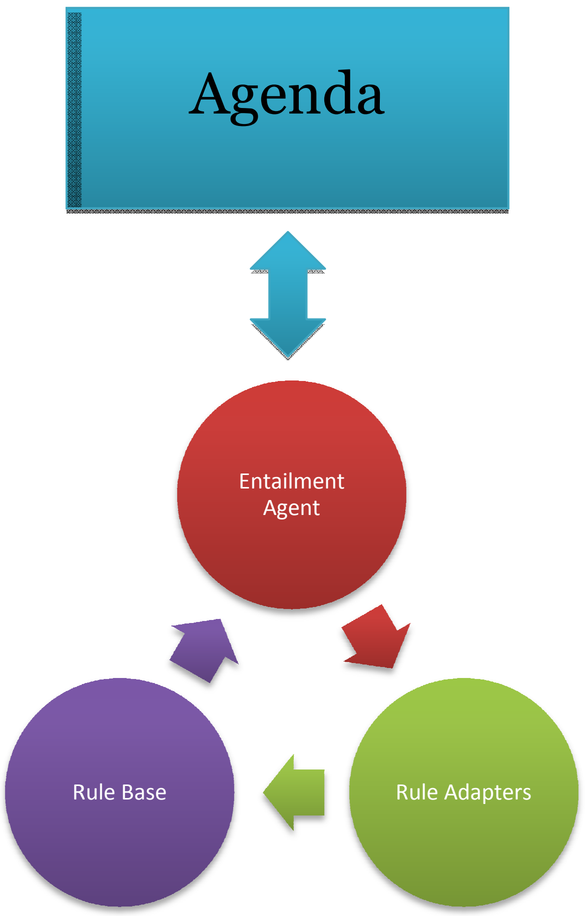


FIGURE 21: REASONING OVER CRISP KNOWLEDGE

The ability of probabilistic inference based on Bayesian reasoning is the key distinctive feature of our reasoner. We designed a class named 'BayesianNetwork' that encapsulates two methods., one for calculating the probability by the Normalisation Rule, and another for calculating Bayes' Rule. Mathematics behind these two methods have already been described in detail (refer to section 2.2.2).

4.5.1 BACKWARD COMPATIBILITY

A good design always provides a bridge for previous versions to adapt the newer ones. Two features make our design backward compatible. Firstly, one should be able to convert from N-Quads to N-Triples; we will describe how this could be done in section 0. Secondly, the reasoner must be able to reason over crisp knowledge. Moreover, the entailment process as it is presented in RDF and RDF Schema should also be present in an N-Quad reasoner. Therefore, a valid N-Quad reasoner is capable of reasoning upon entries that contain elements from RDF and RDF Schema vocabulary.

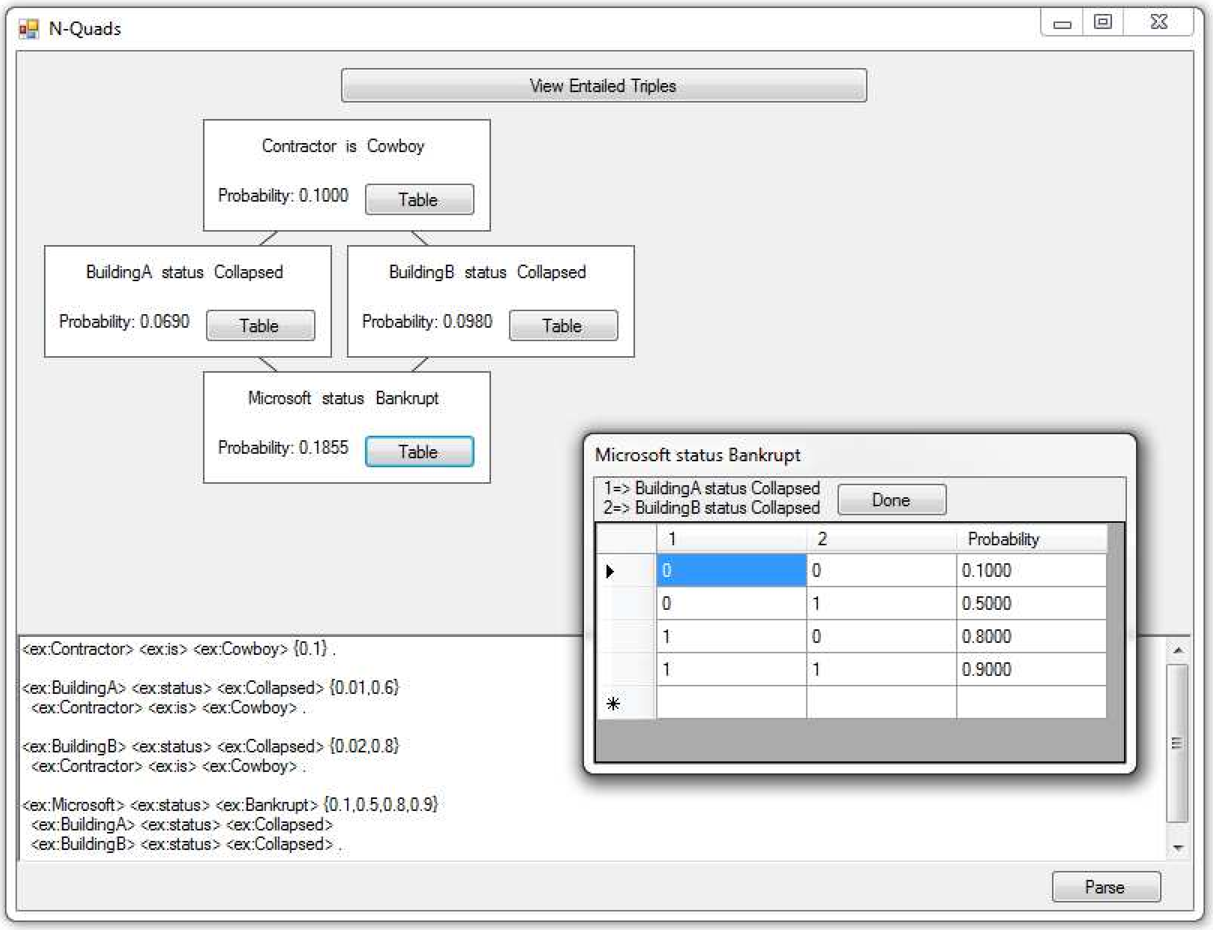
With the aims of supporting inference for RDF and RDF Schema, we implemented the corresponding entailment rules as built in functions of the framework. Mapping from RDF/RDFS entailment rules have been made possible with the aid of our expert system shell in a quite straightforward manner. All we needed to do was to define a class for each of the rules from Table 5 and Table 6 from section 2.3.3.1, and capture the requirements of the rule by simple comparisons of the corresponding elements in the triples (i.e. subject, predicate and object). We have also provided a trace mechanism that enables us to identify from which rule a triple has been entailed.

```
public class Rdfs11 : Rule
{
    public Rdfs11() : base(2) { }

    public override bool Matches(List<SemanticTriple> triples)
    {
        if (triples[0].Predicate.Uri == new Uri("rdfs:subClassOf"))
            if (triples[1].Predicate.Uri == new Uri("rdfs:subClassOf"))
                if (triples[0].Object.Uri == triples[1].Subject.Uri)
                    return true;
        return false;
    }

    public override List<SemanticTriple> ApplyRule(List<SemanticTriple> triples)
    {
        return new List<SemanticTriple>()
        {
            SemanticTriple.FromSPO(triples[0].Subject.Uri,
            new Uri("rdfs:subClassOf"), triples[1].Object.Uri)
        };
    }
}
```

FIGURE 22: IMPLEMENTATION OF RDFS11

4.6 USER INTERFACE

We designed a graphical user interface shown in Figure 23 that allows users to encode the knowledge with N-Quads syntax. A code block is provided for typing the code, from there, a user can click the 'Parse' button to see the topology of the network and results of reasoning.

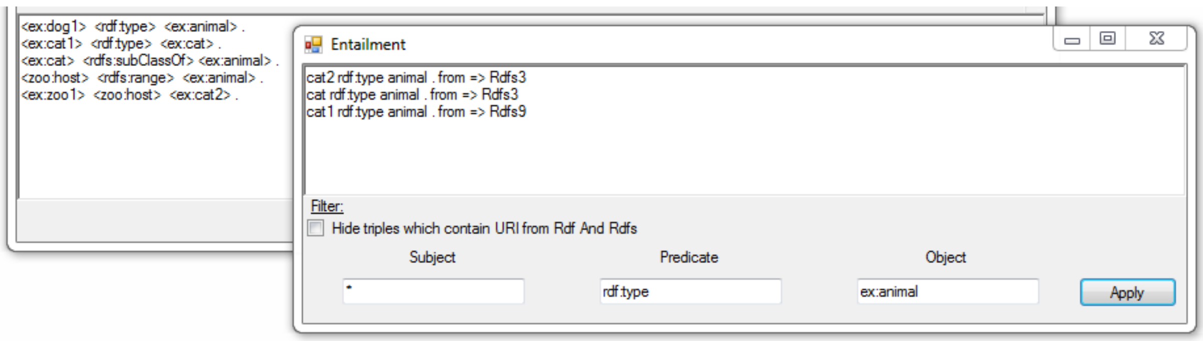


N-Quads
View Entailed Triples
Contractor is Cowboy
Probability: 0.1000
Table
BuildingA status Collapsed
Probability: 0.0690
Table
BuildingB status Collapsed
Probability: 0.0980
Table
Microsoft status Bankrupt
Probability: 0.1855
Table
Microsoft status Bankrupt
1=> BuildingA status Collapsed
2=> BuildingB status Collapsed
Done
Parse

	1	2	Probability
▶	0	0	0.1000
	0	1	0.5000
	1	0	0.8000
	1	1	0.9000
*			

```
<ex:Contractor> <ex:is> <ex:Cowboy> {0.1} .

<ex:BuildingA> <ex:status> <ex:Collapsed> {0.01,0.6}
  <ex:Contractor> <ex:is> <ex:Cowboy> .

<ex:BuildingB> <ex:status> <ex:Collapsed> {0.02,0.8}
  <ex:Contractor> <ex:is> <ex:Cowboy> .

<ex:Microsoft> <ex:status> <ex:Bankrupt> {0.1,0.5,0.8,0.9}
  <ex:BuildingA> <ex:status> <ex:Collapsed>
  <ex:BuildingB> <ex:status> <ex:Collapsed> .
```

FIGURE 23: USER INTERFACE



```
<ex:dog1> <rdf:type> <ex:animal> .
<ex:cat1> <rdf:type> <ex:cat> .
<ex:cat> <rdfs:subClassOf> <ex:animal> .
<zoo:host> <rdfs:range> <ex:animal> .
<ex:zoo1> <zoo:host> <ex:cat2> .
```

```
cat2 rdf:type animal . from => Rdfs3
cat rdf:type animal . from => Rdfs3
cat1 rdf:type animal . from => Rdfs9
```

Subject: *
Predicate: rdf:type
Object: ex:animal

FIGURE 24: RDF AND RDFS REASONING

As Figure 24 indicates, we have provided the users with an interface for applying basic filters on the results of entailment.

5 EVALUATION AND TESTING

In this section, we conduct a qualitative and a quantitative evaluation on the concept of Quads. With respect to the quality of the work, we claim that all of the requirements described in section 3.2 are met. This claim can be supported by testing our design against the user scenarios described in section 3.1:

1. A user can encode the knowledge in the form of crisp rules and reason about it. For the first scenario, Fred can apply our crisp reasoning and receive the results of entailment. A simple SPARQL query will show the results that he requires.
2. The second scenario can be satisfied by converting the knowledge to belief Quads. We argue that N-Quads is the only language that is capable of addressing this domain of users.
3. The third scenario is the most important one. We have conducted a qualitative evaluation for this domain that will be discussed in section 5.2.

5.1 SYNTAX

Using N-Quads' syntax has some advantages and drawbacks. On the plus side, a triple can be encoded with few language elements, thanks to heritage in N-Triples. This feature provides a straightforward mapping from visual triples to code for those who are not coming from a Semantic Web background. Also inheriting from N-Triples is that repeated facts do not count, with the slight difference of the uncertainty value (which is new in N-Quads) to be computed as the maximum value of all repeated entries. Another advantage of the proposed syntax is that an N-Quad file can be converted to N-Triples with minimal processing by a simple rule:

Conversion rule: For every entry in an N-Quad file, if all characters from '{' to '.' are discarded (including the brace, but not the dot), the result will be a valid N-triple file.

This enables us to provide the simplest possible conversion from N-Quads to N-Triples, which does not involve any grammatical or semantic analysis over the syntax (i.e. satisfies the requirement of providing the basis for implementation to be simple). To sum up the advantages, this design for the syntax is simple, expressive, understandable and easy to implement (the ease of deploying proposed design will be demonstrated in coming sections).

The restrictions that we put on the design will instigate some issues with N-Quads. Unfortunately, there are a number of limitations that cannot be worked around without practically losing the simplicity of the design and making the syntax intangible. Top of the list is inability to represent collections. Although the list of givens are clustered within the latter part of an entry, this cannot be considered as a collection since the order in which they appear is critical. In addition, we can only specify a single proposition for each entry, which means if we had two propositions that shared the given part, we would need to allocate two entries for them that were identical in given part, but differed in proposition. In contrast, languages such as N3 are able of representing collections and therefore, do not have issues caused by not having them, such as redundancy in code and overhead for parsing.

Another problem is having the givens as triples instead of references. This will add redundancy to the code (since we most likely have them specified elsewhere) and the syntax will not be intuitively understandable. Perhaps the most irritating disadvantage that can be found in the syntax is that the givens and probability values are not independent. For example, if we had been supplied with three entries as givens, we would have needed to specify corresponding probabilities (eight values) in a way that the order of probability values was concordant with respect to the givens. Table 9 illustrates an example of two possible arrangements for the same Quad and their effect on the order of probability values. As a result, any change in the order of givens (e.g. by placing a new triple in the middle of two others) must be followed by modifying the arrangement of probability values. Therefore, it can be drawn that a modification in the givens part can result in the side effect of allocating new probabilities (likely incorrect ones) to all triples in the given part, if applied incorrectly. This makes it difficult for a human operator to change the entries of a Quad without mistakenly corrupting the dependencies within that Quad.

`<ex:Temperature> <ex:is> <ex:Low>`	`<ex:Temperature> <ex:is> <ex:Low>`
`    {0.1, 0.3, 0.5 ,0.8}`	`    {0.1, 0.5, 0.3 ,0.8}`
`    <ex:Weather> <ex:is> <ex:Rainy>`	`    <ex:Season> <ex:is> <ex:Winter>`
`    <ex:Season> <ex:is> <ex:Winter> .`	`    <ex:Weather> <ex:is> <ex:Rainy> .`

TABLE 9: TWO ARRANGEMENTS FOR THE SAME QUAD

With the purpose of providing a solution for the third user scenario described in section 3.1.1, we need to encode our knowledge in a way that translates to Mr. Richardson's situation. The relations knowledge that fictional Mr. Richardson had provided us with are intentionally the same as the relations in Microsoft's Buildings exampled described in section 2.2.2. For the sake of fewer calculations, let us assume that the probabilities we came to acquire are too the same. The Bayesian Network which we are dealing with thereby, will match Figure 4 except for naming of the nodes. An advantage of N-Quads is its ability to express the probabilistic relations in knowledge from different domains that would not be possible to encode in the form of crisp triples. This has been illustrated in Figure 25.

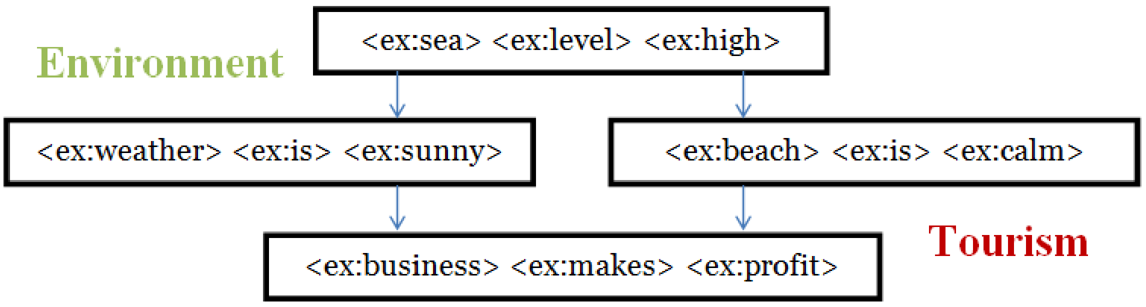

Environment
<ex:sea> <ex:level> <ex:high>
<ex:weather> <ex:is> <ex:sunny>
<ex:beach> <ex:is> <ex:calm>
<ex:business> <ex:makes> <ex:profit>
Tourism

FIGURE 25: KNOWLEDGE FROM TWO DIFFERENT DOMAINS MET IN ONE PROBLEM

Our goal is to show the difference of the case that one takes the advantage of potential knowledge behind the relations in a domain, and the case where one does not. Accordingly, we designed two software agents to predict if the business makes profit the next day; one working only based on the probability of bottom node, which we called 'Believer', and the other one armed with Bayesian Reasoning abilities of N-Quads, who we named 'Flexi' (by which we mean it represents knowledge in a flexible manner). The task here is to test the reasoning ability of N-Quads in the domain that Mr. Richardson has specified. We accomplished this task by designing a simulation environment. Below are the specifications of the simulation:

P(weather is sunny) = 0.069 , P(sea level high) = 0.1 , P(beach is calm) = 0.098 ,
P(business makes profit) = 0.1855 , Time measure = day
Correct prediction = 4 utility points , Incorrect prediction = -4 utility points

The figure below shows the results from a sample run:

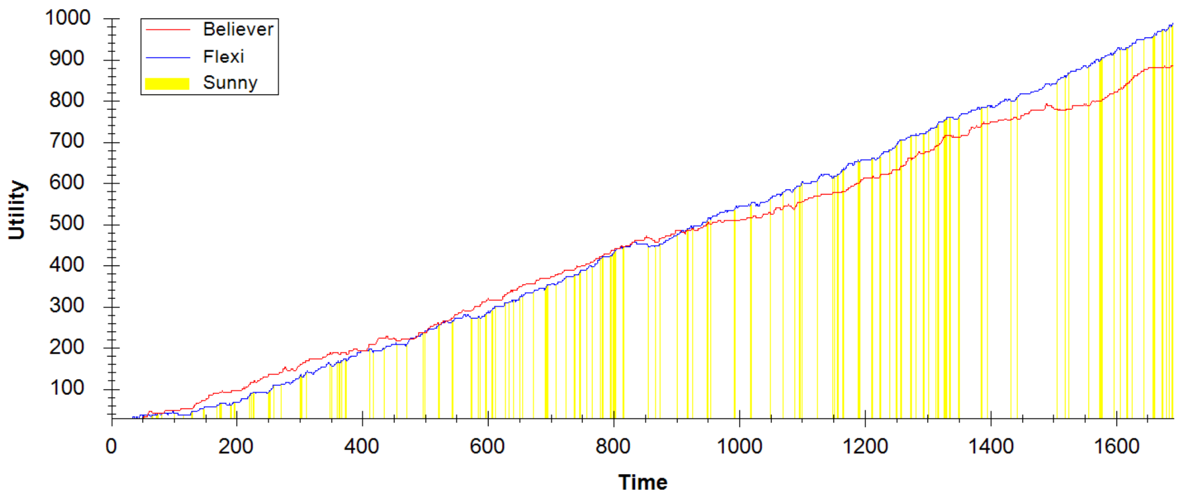


FIGURE 26: FLEXI VS. BELIEVER, PARTIAL GRAPH

The results from Figure 26 can be misleading to the conclusion that the performance of Flexi is not much better than Believer. In fact, Flexi even loses against Believer in the first few hundred time steps. However, by allowing the same run to continue for a longer time, we got the results shown in Figure 27.

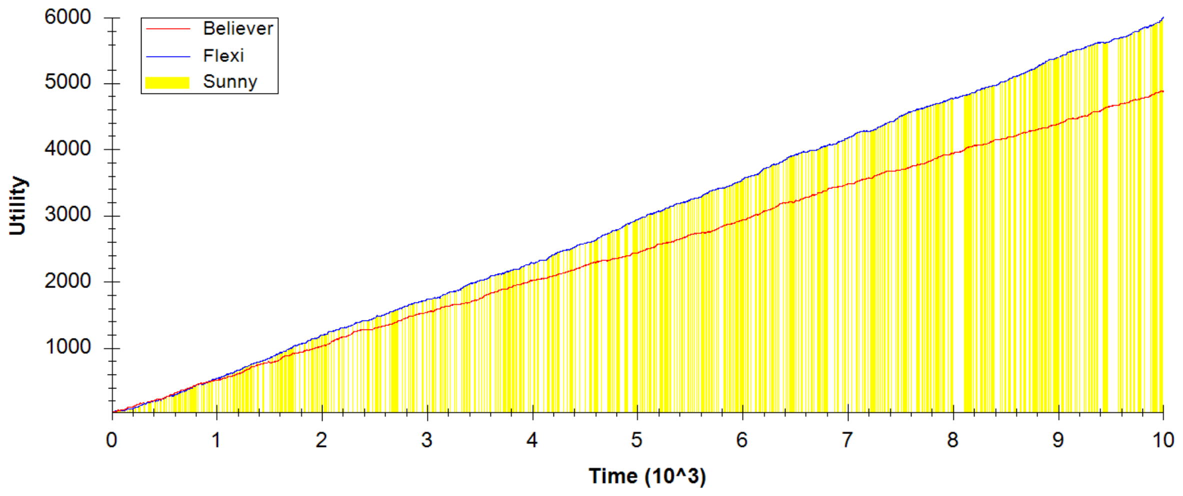


FIGURE 27: FLEXI VS. BELIEVER, ENTIRE GRAPH[1]

We can identify that Flexi is linearly superior, but this is only by having one given parameter; it could even increase exponentially if we had more information about the domain specified (which would be one of the few examples where something being exponential has worked for computer science). The element that changes the game in favor of Flexi is the sun. For each time step in which the weather is sunny, Flexi will have a better estimation of probabilities than Believer. Imagine that each utility point was worth 200 units of money. In 1500 time steps, Flexi has some 200 utility points more as compared to Believer. This suggests that relational probabilities are more effective than pure beliefs (Believer considers the uncertainty of the domain). By simple calculations, Flexi could earn us 200,000 units in the period of approximately four years. This amount of money will be more than enough for saving Mr. Richardson's business from bankruptcy. Now imagine a web in the scale of the world, having its relational uncertainties annotated by millions of users, the achievements would be endless.

[1] Our plotting software is imperfect, which is the reason that the frequency of sunny weather appears to be more in Figure 27 than in Figure 26. Readers can assume a density of 0.069 for yellow bars in general, according to calculations from section 2.2.2.

6 CONCLUSION AND FUTURE WORK

This dissertation reviewed different approaches for integrating uncertainty within the Semantic Web. We showed that considering uncertainty in the process of reasoning could have a substantial impact on the robustness of Semantic Web applications. A number of scenarios have been presented that could benefit from probabilistic inference. We introduced the concept of N-Quads, which we argued is potentially the best way of fitting uncertainty into the Semantic Web, then we checked the test scenarios against the hypothesis. Finally, we evaluated our approach, both qualitatively and quantitatively. The results have shown that the idea of shifting from ontology level to language level can be both effective and useful. Therefore, dedicating future research on the concept of Quads will certainly be worth it.

Future works can investigate within the following categories:

1. Adding support for variables to N-Quads: Crisp rules can currently be hard-coded by the tools that we developed in this work. An example is the entailment rules from RDF and RDF Schema, which are already in place at the current implementation. A possible future direction is to extend the syntax for supporting variables. This will advance N-Quads in two ways. Firstly, it will make it possible to define the crisp rules by the syntax itself instead of having to hard-code them. Secondly, uncertainty values can be variables that one can calculate from other pieces of knowledge in the domain.

2. Building a scale: The current proposal of N-Quads is just a proof of concept that demonstrates the possibility of shifting uncertainty handling from ontology level to language level. Research could be done in the areas of syntax and efficient reasoning.

3. Automated process of finding relational probabilities: The uncertainty values can be computed in linear time complexity by conducting simple statistical surveys. It would be useful if one could present a method for doing this in a systematic way.

We would like to conclude the dissertation by emphesising on the importance of uncertainty handling for the Semantic Web. Current Semantic Web standards are more focused towards pure logic. However, a world of knowledge is hidden behind probabilities that could be utilised if in favour of knowledge management.

BIBLIOGRAPHY

BERNERS-LEE, T. 1980. *The ENQUIRE System* [Online]. W3C. Available: http://www.w3.org/History/1980/Enquire/manual/ [Accessed 10-12-2011].

BERNERS-LEE, T. 1989. *Information Management: A Proposal* [Online]. W3C. Available: http://www.w3.org/History/1989/proposal.html [Accessed 11-11-2011].

BERNERS-LEE, T. 1999. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*, Harper San Francisco.

BERNERS-LEE, T. 2006. *Linked Data* [Online]. W3C. Available: http://www.w3.org/DesignIssues/LinkedData.html [Accessed].

BRACHMAN, R. J., MCGUINNESS, D. L., PATEL-SCHNEIDER, P. F., RESNICK, L. A. & BORGIDA, A. 1991. Living with CLASSIC: When and how to use a KL-ONE-like language. *Principles of semantic networks*, 401-456.

BRACHMAN, R. J. & SCHMOLZE, J. G. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive science,* 9**,** 171-216.

BRICKLEY, D. & MILLER, L. 2005. *FOAF Vocabulary Specification* [Online]. Available: http://xmlns.com/foaf/spec/20050403.html [Accessed 10-1-2012].

BRIN, S. & PAGE, L. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems,* 30**,** 107-117.

BRONOWSKI, J. 1974. *The ascent of man*, BBC Books.

BUSH, V. 1945. As we may think. *The Atlantic Monthly (July 1945)*.

CARMODY, S., GROSS, W., NELSON, T. H., RICE, D. & VAN DAM, A. 1969. A hypertext editing system for the /360. *In:* FAIMAN & NIEVERGELT (eds.) *Pertinent Concepts in Computer Graphics.* Proceedings of the Second University of Illinois Conference on Computer Graphics.

COLLINS, A. M. & QUILLIAN, M. R. 1969. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior,* 8**,** 240-247.

COSTA, P. C. G., LASKEY, K. B. & LASKEY, K. J. 2008. PR-OWL: A Bayesian ontology language for the semantic web. *In:* FANIZZI, N., AMATO, C., LASKEY, K. B., LASKEY, K. J., LUKASIEWICS, T., NICKLES, M. & POOL, M. (eds.) *Uncertainty reasoning for the semantic web I.* Springer.

DEROSE, S. J. & VAN DAM, A. 1999. Document structure and markup in the FRESS hypertext system. *Markup Languages,* 1**,** 7-32.

DING, Z. & PENG, Y. Year. A probabilistic extension to ontology language OWL. *In:* Proceedings of the 37th Hawaii International Conference on System Sciences, 2004. IEEE, 10 pp.

DINUCCI, D. 1999. Fragmented Future. *Print,* 53**,** 220-222.

ENGELBART, D. C. & ENGLISH, W. K. Year. A research center for augmenting human intellect. *In*, 1968. ACM, 395-410.

FENSEL, D. & VAN HARMELEN, F. 2007. Unifying reasoning and search to web scale. *IEEE Internet Computing***,** 96.

FENSEL, D., VAN HARMELEN, F., HORROCKS, I., MCGUINNESS, D. L. & PATEL-SCHNEIDER, P. F. 2001. OIL: An ontology infrastructure for the semantic web. *Intelligent Systems, IEEE,* 16**,** 38-45.

FLORIDI, L. 2009. Web 2.0 vs. the Semantic Web: A Philosophical Assessment. *Episteme,* 6**,** 25-37.

GOLBECK, J., PARSIA, B. & HENDLER, J. Year. Trust Networks on the Semantic Web. *In:* Proceedings of Cooperative Intelligent Agents, 2003. 238-249.

GRANT, J. & BECKETT, D. 2001. *RDF Test Cases* [Online]. Available: http://www.w3.org/TR/rdf-testcases/ [Accessed].

GUHA, R. V. 1997. Meta Content Framework. Apple Computer.

HALPERN, J. Y. 2005. *Reasoning about Uncertainty*, The MIT Press.

HAYES, P. 2004. *RDF Semantics* [Online]. W3C. Available: http://www.w3.org/TR/rdf-mt/ [Accessed 01-11-2011].

HENDLER, J. & MCGUINNESS, D. L. 2000. The DARPA agent markup language. *IEEE Intelligent systems,* 15**,** 67-73.

HERMAN, I. 2007. State of the Semantic Web. *Semantic Days.* Stavanger, Norway.

HORROCKS, I. 2002. DAML+OIL: A Description Logic for the Semantic Web. *IEEE Data Engineering Bulletin,* 25**,** 4-9.

KLYNE, G. & CARROLL, J. J. 2004. *Resource Description Framework (RDF)* [Online]. W3C. Available: http://www.w3.org/TR/rdf-concepts/ [Accessed 01-11-2011].

KNORR, E. 2003. *2004:The year of Web service* [Online]. CIO. Available: http://www.cio.com/article/32050/2004_The_Year_of_Web_Services [Accessed 20-11-2011].

KORB, K. B. & NICHOLSON, A. E. 2004. *Bayesian Artificial Intelligence*, Chapman & Hall/CRC.

LACY, L. W. 2005. *OWL: Representing information using the web ontology language*, Trafford Publishing.

LASSILA, O. & SWICK, R. R. 1999. *Resource Description Framework (RDF) Model and Syntax Specification* [Online]. W3C. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ [Accessed 5-1-2012].

LUKASIEWICZ, T. 2007. Probabilistic Description Logics for the Semantic Web. INFSYS.

MCCULLOCH, W. S. & PITTS, W. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology,* 5**,** 115-133.

MINSKY, M. 1974. A Framework for Representing Knowledge. *MIT-AI Laboratory Memo 306*.

MOTIK, B., GRAU, B. C., HORROCKS, I., WU, Z., FOKOUE, A. & LUTZ, C. 2009. *OWL 2 Web Ontology Language Profiles* [Online]. W3C. Available: http://www.w3.org/TR/owl2-profiles/ [Accessed].

NARDI, D. & BRACHMAN, R. J. 2003. An Introduction to Description Logics. *In:* BAADER, F., CALVANESE, D., MCGUINNESS, D., NARDI, D. & PATEL-SCHNEIDER, P. (eds.) *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press.

NELSON, T. H. 1967. Getting it out of our system. *Information retrieval: A critical review***,** 191-210.

NELSON, T. H. 1980. *Literary Machines*, Mindful Press.

NOBLE, J. 2007. Probability Overview. *Bayesian Networks.* Southampton: University of Southampton.

O'REILLY, T. & BATTELLE, J. 2004. Opening Welcome: State of the Internet Industry. *Web 2.0.* San Francisco, California: O'Reilly Media and MediaLive.

O'REILLY, T. 2005. *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software* [Online]. O'Reilly Media. Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-Web-20.html [Accessed 29-11-2011].

PAULOS, J. A. 1988. *Innumeracy: Mathematical illiteracy and its consequences*, Hill & Wang.

QUILLIAN, M. R. 1967. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science,* 12**,** 410-430.

RABIN, M. O. & SCOTT, D. 1959. Finite automata and their decision problems. *IBM journal of research and development,* 3**,** 114-125.

RECTOR, A. & NOWLAN, W. 1994. The GALEN project. *Computer Methods and Programs in Biomedicine,* 45**,** 75-78.

RECTOR, A. L., BECHHOFER, S., GOBLE, C. A., HORROCKS, I., NOWLAN, W. A. & SOLOMON, W. D. 1997. The GRAIL concept modelling language for medical terminology. *Artificial intelligence in medicine,* 9**,** 139-171.

RICHARDSON, M., AGRAWAL, R. & DOMINGOS, P. Year. Trust Management for the SemanticWeb. *In:* Proceedings of the Second International Semantic Web Conference 2003 Richardson, 2003. 351-368.

RUSSELL, S. J. & NORVIG, P. 1998. *Artificial intelligence: A Modern Approach*, Prentice hall.

SCHANK, R. C. 1972. Conceptual dependence: A theory of natural language understanding. *Cognitive Psychology,* 3**,** 552-631.

SCHANK, R. C. & ABELSON, R. P. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*, Lawrence Erlbaum Associates Hillsdale, NJ.

SHADBOLT, N., HALL, W. & BERNERS-LEE, T. 2006. The semantic web revisited. *IEEE Intelligent Systems,* 21**,** 96-101.

SHETH, A., RAMAKRISHNAN, C. & THOMAS, C. 2005. Semantics for the Semantic Web: The Implicit, the Formal and the Powerful. *International Journal on Semantic Web & Information Systems,* 1**,** 1-18.

SHIRKY, C. 2003. *The Semantic Web, Syllogism, and Worldview* [Online]. "Networks, Economics, and Culture" mailing list. Available: http://www.shirky.com/writings/semantic_syllogism.html [Accessed].

SIMON, H. A. & NEWELL, A. 1958. Heuristic problem solving: The next advance in operations research. *Operations research,* 6**,** 1-10.

SMITH, M. K., WELTY, C. & MCGUINNESS, D. L. 2004. *OWL Web Ontology Language Guide* [Online]. W3C. Available: http://www.w3.org/TR/owl-guide/ [Accessed 01-11-2011].

SOWA, J. F. 1983. *Conceptual structures: Information processing in mind and machine*, Addison-Wesley.

STRACCIA, U. 2011. Reasoning within Fuzzy Description Logics. *Journal of Artifcial Intelligence Research,* 14**,** 137-166.

THOMAS, C. & SHETH, A. 2011. Web Wisdom: An Essay on How Web 2.0 and Semantic Web Can Foster a Global Knowledge Society. *In:* TENNYSON, R. (ed.) *Computers in Human Behavior.* Elsevier Ltd.

THOMAS, C. & SHETH, A. P. Year. Semantic convergence of wikipedia articles. *In:* Proceedings of the 2007 IEEE/WIC International Conference on Web Intelligence, 2007. Washington, DC, USA: IEEE Computer Society, 600-606.