Motivation, Possibilities, and Difficulties for Participating in the Semantic Web

Aning Rothe

Professor: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Supervisor: Dipl.-Inf. habil. Marius Feldmann

Department of Computer Science
Institute for System Architecture
TECHNISCHE UNIVERSITÄT DRESDEN
July, 2008
Author: Aning Rothe
Matrikel-Nr.: 2810396
Title: Motivation, Possibilities, and Difficulties for Participating in the Semantic Web
Degree: Master of Science

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INSTITUTE FOR SYSTEM ARCHITECTURE
The author reserves all rights. This work may not be reproduced in whole or in part without the author’s written permission.

Permission is granted to Technische Universität Dresden to make copies for non-commercial purposes.

The author certifies that all works during this research has been completed by myself. Any copyrighted material that appeared in this work has been cited, and all external helps that I received in this research work have been clearly acknowledged.

______________________________
Signature of Author

Dated: ________________________
I would like to express my heartfelt thanks to my supervisor Mr. Marius Feldmann, who gave me the chance to do this thesis. Without his advice, encouragement, enduring patience and constant support, the work presented in this thesis would not have been possible.

I also wish to thank Prof. Dr. Alexander Schill who provided direction and all ideas along the way as I expected. I appreciate his guidance and additional helpful proposal I needed in order to accomplish this thesis.

Additionally, I am greatly indebted to Liu, Wu, Deng and Wason for sharing and providing me their knowledge, and to Xu for giving her advice. In particular Adeles and Danny, who gave up their valuable time to proofread the drafts with incredible care and to offer comments that have led to many improvements. Of course, other friends cannot go unmentioned: Joey, Echo, and Yan.

A special mention must go to my father, who was my only family member who thought that I must go on with study for Master degree after giving birth to my daughter.

Finally, I must thank my dear parents and parents-in-law, my dear husband and my dear daughter. They were always willing to help in any way they could, and their love is very much appreciated.
Acknowledgements
Abstract

The Semantic Web was widely adopted as a vision and a challenge in our days. It has been regarded as the next generation of the World Wide Web. The Semantic Web extend the current World Wide Web so that realize the full potential of the Web by enabling software agents on the Web to understand and process information autonomously and collaboratively. Hence the Semantic Web is becoming a key element for changing our familiar internet. However, the current Web is far away from becoming a network of computer processable information.

Due to this above fact, this thesis provides an analysis of the status quo of the Semantic Web research and development out of a practical view, and then concentrates on practical perspective of introducing how to add semantics to existing Web content. This thesis also presents a fundamental understanding of available ontology tools on the market in terms of their capability, availability and further development. Furthermore, it presents a specific application of practice for transforming a Web application to a system of semantically annotated information, and corresponding applied implementation tools. Based on this specific application, a useful strategy for transforming a Web application to the Semantic Web is proposed.

This thesis constitutes an advance in the direction for participating in the Semantic Web. It provides an evaluation of most common and widespread implementation tools for applications. It should, therefore, be of foundation for further investigations of the Semantic Web and, in general, ontology applications.
Acronyms

\( \mathcal{ALC} \) : Attributive language with complement (full concept negation)

\( \mathcal{ALC}(\mathcal{D}) \) : \( \mathcal{ALC} \) extended with a concrete domain \( \mathcal{D} \)

DAML : DARPA Agent Markup Language

DAML+OIL : the result of merging DAML and OIL

DL : Description Logic

HTML : Hypertext Markup Language

OIL : Ontology Inference Layer

OWL : Web Ontology Language, a W3C recommendation

\( \text{OWL 1.1} \) : a revision of OWL DL, which plus qualified cardinality restrictions \( Q \); local reflexivity restrictions for simple properties; reflexive, irreflexive, symmetric, and asymmetric properties; disjoint simple properties; and property chain inclusion axioms, based on the logic language \( SROIQ \)

RDF : Resource Description Framework

RDFS : RDF Schema

\( S \) : \( \mathcal{ALC}_{R^+} \), \( \mathcal{ALC} \) extended with transitive role axioms(\( R^+ \))

\( SH \) : \( S \) extended with role hierarchy (\( H \))

\( SHI \) : \( SH \) extended with inverse roles (\( I \))
Acronyms

\textit{SHIO} : \textit{SHI} extended with nominals (\textit{O})

\textit{SHIQ} : \textit{SHQ} extended with inverse roles (\textit{I})

\textit{SHOIN} : \textit{SHIO} extended with number restriction (\textit{N})

\textit{SHOIQ} : \textit{SHIQ} extended with nominals (\textit{O})

\textit{SHQ} : \textit{SH} extended with qualified number restrictions (\textit{Q})

\textit{SR} : \textit{SH} extended with

\textit{SW} : Semantic Web

\textit{XML} : eXtensible Markup Language

\textit{W3C} : World Wide Web consortium
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acronyms</td>
<td>v</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Current Status of the Web</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Structure of the Thesis</td>
<td>2</td>
</tr>
<tr>
<td>2 State-of-the-art</td>
<td>5</td>
</tr>
<tr>
<td>2.1 The World Wide Web</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Ajax</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 The Social Web</td>
<td>10</td>
</tr>
<tr>
<td>2.2 The Semantic Web</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Languages in the Semantic Web</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2 Description Logics and the Semantic Web</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3 Tools in the Semantic Web</td>
<td>20</td>
</tr>
<tr>
<td>2.2.4 Current Research on the Semantic Web</td>
<td>24</td>
</tr>
<tr>
<td>3 Motivation for Participating in the Semantic Web</td>
<td>29</td>
</tr>
<tr>
<td>3.1 General Aspects</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Motivating Scenarios</td>
<td>30</td>
</tr>
<tr>
<td>3.2.1 Intelligent Search Scenario</td>
<td>30</td>
</tr>
<tr>
<td>3.2.2 Medical Semantic Scenario</td>
<td>32</td>
</tr>
<tr>
<td>3.2.3 An E-learning Scenario</td>
<td>32</td>
</tr>
</tbody>
</table>
4 Implementation Details

4.1 Transforming a Web Application to the Semantic Web ............ 35
   4.1.1 Ontology Creation ........................................... 35
   4.1.2 Decision Regarding Tools ................................. 37
   4.1.3 Realization based on Semantic Annotation .................... 39

4.2 General Strategy ................................................... 40

4.3 Summary ............................................................. 42

5 Evaluation ............................................................... 43

5.1 Protégé vs SMORE ................................................... 43

5.2 Evaluation of Reasoners ............................................. 44

6 Conclusions and Outlook ............................................. 49

Bibliography ............................................................... 51
Chapter 1

Introduction

1.1 Current Status of the Web

The World Wide Web\(^1\), commonly shortened to “the Web” or “WWW”, is global information medium where data is shared mainly for human consumption. People could make use of information for seeking, getting in touch with each other, and ordering products by filling out forms online. Search engines are main valuable tools for using today’s Web that are made up by languages such as HTML, CSS, etc. In spite of some improvements in search engine technology during past few years, problems still appeared. The amount of Web content outpaces technological progress. Information can not be easily understood and processed by machines. This fact limited the usefulness of the Web.

The vision of the Semantic Web is to augment the current World Wide Web so that Web resources (i.e., Web pages as well as a wide range of Web accessible data and services) are given explicit, well-defined, content-related meaning. The Semantic Web (SW) aims at enabling software agents (intelligent software) to automatically process and integrate information available from disparate sources on the Web. The World Wide Web Consortium (W3C) has developed a set of standards and tools to support the Semantic Web. However, people are still asking how the Semantic Web can be used in practical situations to solve real problems.

\(^1\)Please visit http://www.w3.org/WWW/ for more details.
Therefore, realization of the Semantic Web becomes an enthusiastic topic for researchers. The realization of the Semantic Web depends on the ability to semantically markup Web resources through annotations which are machine-understandable. Description logics are often named as one of the tools that help to realize the Semantic Web. They are very useful for defining, integrating, and maintaining ontologies. Ontologies are explicit specifications of conceptualizations; they establish a joint terminology and make sure that different software agents have a common understanding about certain annotation of the Web page.

The use of ontologies in applications requires tools. At present, large numbers of development tools, program libraries and environments have emerged to support the development of the Semantic Web.

1.2 Structure of the Thesis

The rest of this thesis is structured as follows. In Chapter 2, we detail the current status of main research areas in the field of the World Wide Web and the Semantic Web. In Section 2.1 the World Wide Web is introduced simply including Ajax and Social Web 2.0. In Section 2.2 we give a brief account of Semantic Web various languages, notations, techniques and tools.

Motivation for participating in the Semantic Web is introduced in Chapter 3. In Section 3.1 we first introduce the general motivation. In Section 3.2 we will present three realistic scenarios as examples to show why we need to participate in the Semantic Web.

Implementation takes the central stage in Chapter 4 in this thesis. We will explore some issues when developing Semantic Web applications. Since this field is rather new, few people have experience in the development of real-world systems. Thus, it is important to collect examples during application scenarios that illustrate problems and challenges. We illustrate one specific scenario connected to a news portal, and suggest a step-by-step strategy for developing Semantic Web
applications. From this strategy we will derive some development guidelines.

The most common and widespread software components, such as Jena, Racer, and Pellet and so on, are evaluated in Chapter 5. It also contains an analysis of the effect of the semantic application implementation tools on practice. Chapter 6 concludes the thesis, summarizes the stumbling blocks and bottlenecks of participating in the Semantic Web, and discusses directions of future work.
Chapter 1. Introduction
Chapter 2

State-of-the-art

The Internet has been integrated into a lot of people’s lives. During the past two decades, the world has witnessed a Web technological evolution that is provided through two stages of transition. One stage started with static Web sites at the beginning and then moved on to dynamic Web sites. Another stage has already started with syntactical Web sites and then to Semantic Web sites. Following these two stages, a clear and convincible line of Web developments will be introduced in details.

2.1 The World Wide Web

The World Wide Web was first proposed by the British Tim Berners-Lee, in the European Community in 1989. At that time, this concept has been quite simple: it used the hypertext documents (e.g. HTML\footnote{http://www.w3.org/html/}) and URIs (Uniform Resource Identifiers) \cite{1} to create a network of interconnected information. Through the Web, internet resources can be linked to each other. For example, it connects to various academic papers from parts of the world so that people can immediate access to the useful materials. The World Wide Web is thus a universal network of Web sites. But it is just a platform for sharing the syntactical resources and not for the semantically annotated resources. The traditional World Wide Web development was started from such static pages.
Chapter 2. State-of-the-art

The traditional World Wide Web has often been named “Web 1.0” since the hype of “Web 2.0”. Web 1.0 refers to the collection of Web site applications before the Web 2.0 craze, namely, between in the period 1994 and 2004. In fact, Web 1.0 is one buzzword. There are no defined boundaries between Web 1.0 and Web 2.0. Essentially, Web 1.0 is for humans to read and understand. There is no obligation of machine process ability. On Web 1.0, webmasters write down what they want to share and then publish it online, web readers can well understand their contents. But no feedback messages could be delivered back to webmasters from readers, unless webmasters release their contact information in their Web pages. The link between webmasters and web readers is generally not so strong on Web 1.0.

The companies Netscape, Yahoo and Google made tremendous contributions for Web 1.0. Netscape developed the first large-scale commercial browser. Yahoo proposed the internet yellow pages that classified the internet. Google recommended a popular search service [2].

On the Web, when the amount of Web resources becomes too many to be efficiently operated by the mechanisms at the meantime, new mechanisms must be formed to handle this quantitative accumulation. Therefore, one stage transition occurs, that from Web 1.0 to Web 2.0.

The term “Web 2.0” was coined by Dale Dougherty of O’Reilly Media brainstorming with Craig Cline of Medialive to develop ideas for a conference that they could jointly host. A lot of people asked about what Web 2.0 is. Web 2.0 can be a read and write Web. In some cases it supports not only webmasters but also readers to read and write to the same Web page. This advance allows web users to establish friendly social communication with each other. Hence it increases the participation interest of web users. Normal web readers don’t need to release their private identities prior to publish their viewpoints. The Web 2.0 describes technological changes and social changes in the Web during the past few years.

Web 2.0 is a new generation model of the internet that is based on a new set
2.1. The World Wide Web

of technologies such as Ajax. Hence, there is theoretical advancement in Web 2.0. Web 2.0 describes new forms of applications and services that are accessible by the Web browser. Blog\(^2\), RSS\(^3\), Wiki\(^4\) and other social software applications are the success of Web 2.0. Flickr and YouTube\(^5\) are typical representatives of Web 2.0.

Web 2.0 consists of three parts: The first part is Rich Internet Application\(^6\), some buzzwords that related to that application are Flash\(^7\) and Ajax. Rich Internet Application (RIA) permits us to bring the experience from the desktop into the browsers where we form a graphical point of view or form the use ability point of view, such as drag and drop, which are always used in the desktop. Ajax technology is the most popular technology, which greatly improves user experience of the traditional Web applications, is also known as the traditional Web technology revolution. At the moment, the term Ajax is used very often, because it has the W3C standards and be fully supported by reality browser. Thus it became the most representative and reality-based solution of Rich Internet Applications. We will introduce Ajax in more detail in section 2.1.1.

The second part of Web 2.0 are Service-based systems. It is one of the key parts of Web 2.0 that include such as Feeds\(^8\), e.g., RSS. A Service-based system is about how Web 2.0 applications expose their functionality so that other applications can ever join into the functionality, and provide a much richer set of applications.

The third part of Web 2.0 are Social Web\(^9\). Web 2.0 applications tend to interact much more with the end-user. The end-user is not only a user of the application, but is also a participant that is contriving to the Wiki or doing pod-

\(^2\)http://en.wikipedia.org/wiki/Blog  
\(^3\)http://www.w3schools.com/rss/default.asp  
\(^4\)http://en.wikipedia.org/wiki/Wiki  
\(^5\)http://youtube.com  
\(^6\)http://en.wikipedia.org/wiki/Rich_Internet_application  
\(^7\)http://www.w3schools.com/Flash/default.asp  
\(^8\)http://www.sixapart.com/about/feeds  
\(^9\)http://en.wikipedia.org/wiki/Social_web
cast\textsuperscript{10} through blogging. It is the social nature of these applications, providing feedback, and allows the application to learn from the user through the user using it.

2.1.1 Ajax

Ajax \textsuperscript{3} is a key component of Web 2.0 applications. This name was proposed in a seminal essay by Jesse James Garrett of Web design firm Adaptive Path. It is shorthand for Asynchronous JavaScript and XML. Ajax represents a fundamental shift in what is possible on the Web. Jesse James Garrett wrote: “Ajax is not a technology. It is really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:

- standards-based presentation using XHTML\textsuperscript{11} and CSS\textsuperscript{12};
- dynamic display and interaction using the Document Object Model (DOM)\textsuperscript{13};
- data interchange and manipulation using XML\textsuperscript{14} and XSLT\textsuperscript{15};
- asynchronous data retrieval using XMLHttpRequest\textsuperscript{16};
- and JavaScript\textsuperscript{17} binding everything together.”

The Figure 2.1 shows the distinction between the traditional Web application model and the Ajax Web application model \textsuperscript{3}. The classic Web applications make a lot of technical sense, but while the server processes requests, users must wait at every step in a task. An Ajax approach for applications would not make users waiting for the server to do something by an Ajax engine between the user and the server. Obviously, the fundamental interaction model of the Web is changed by Ajax. The Ajax engine makes the application process very natural and its

\textsuperscript{10}http://en.wikipedia.org/wiki/Podcast
\textsuperscript{11}See HTML http://www.w3.org/Consortium/siteindex#HTML
\textsuperscript{12}http://www.w3.org/Style/CSS/
\textsuperscript{13}http://www.w3.org/DOM/
\textsuperscript{14}http://www.w3.org/XML/
\textsuperscript{15}http://www.w3.org/TR/1999/REC-xslt-19991116
\textsuperscript{16}http://en.wikipedia.org/wiki/XMLHttpRequest
\textsuperscript{17}http://www.w3schools.com/JS/default.asp
2.1. The World Wide Web

Figure 2.1: The traditional model for Web applications compared to the Ajax model

operation is very smooth, because it only exchanges updated data with the server, and unnecessary data for display pages is no longer reloaded. In fact, the Ajax engine is the comprehensive application of the technology of JavaScript, XML and XMLHttpRequest, etc.

Google Maps and Gmail are two examples of this new approach to Web applications. Google Maps used Ajax to enhance maps application, and allows user to drag maps in different directions. Maps are broken down into a group of images. They constitute together as a continuous image. When a user drags maps, it does not need to create a new image, but it shows impact for user that seemingly unlimited rolling. Figure 2.2 shows the Google Maps interface.

Gmail has already provided some unprecedented innovations in email. It was introduced by Google in May 2004. Gmail builds on the idea that users never have to delete mails and users always be able to find the message they want. Gmail also can use Google search to find the exact message users want, no matter when it was sent or received. The screenshot of Gmail’s home page interface depicted in the following Figure 2.3 shows some of the new features such as search, conversation, and email starting.
2.1.2 The Social Web

When we discuss the Web 2.0, we mostly tend to focus on the technologies and not as much on the social phenomena underlying Web 2.0 applications. However, in the past few years it has become increasingly clear with the growth of sites such as MySpace and Facebook. Before we discussed Ajax technology, Google Maps is considered the prototypical Web 2.0 application, but it does not include interaction between users. The idea that users can create content is considered a critical aspect of Web 2.0. Blog and Wikipedia focus on this aspect. The social Web sites primarily allow linking between people, not content, and its networking construction is a key to the success of Web 2.0 applications. Jimmy Wales who is developer of Wikipedia stated in his talk at the Doors of Perception Conference in 2005: “Wikipedia is not primarily a technological innovation, but a social and design innovation”.

Wikipedia\textsuperscript{18} is an online free encyclopedia building collaborations. It allows

\textsuperscript{18}http://www.wikipedia.org/
2.1. The World Wide Web

users to freely create and edit Web page content using any Web browser. Basically, a Web site operates on a principle of collaborative trust. A Wikipedia provides a simplified interface. At any time, contributors can conveniently view the Web page as it looks to other subscribers, before and after the changes they have made.

Blogs are considered as another successful modern social Web application. A blog is a personal online journal in reverse chronological order that is frequently updated. The term blog is a shortened form of weblog or web log. The activity of updating a blog is “blogging”. Individual articles on a blog are called “blog post”, “post” or “entries”. Someone who posts these entries is called a “blogger”. Blogs use a conversational style of documentation. Topics focus on the issues that the author favors. Blogs allow people with little or no technical background to update and maintain it daily. After 911 Terrorist Attacks Incident, it was found that the blog, whose keeper has appeared at the scene of terrorist incidents, gave the first-hand information and the primary real situations about event.

RSS stands for both, Rich Site Summary and Really Simple Syndication, but it always refers to the same technological principle. It is a way to easily distribute a list of headlines, update notices, and sometimes contents to a wide number of
people. RSS is used by computer programs that organize the latest available headlines and/or notices for easy reading. For examples, most people are interested in such news websites and product information pages, but those content changes on an unpredictable schedule. RSS provides us a better way to be aware of news and changed content. It is a mean of transmitting and updating news in an automated way, and maintains a list of notifications on their website in a standard way. This list of notifications is called an “RSS Feed”. People can check this list and find out the latest headlines. RSS aggregators are the special computer programs, it automatically accesses the RSS feeds of website users care about and organizes the results for users.

Today’s Web has become a vastly powerful tool for communication, research and commerce. It is usually presented in databases, and most of current Web content is suitable for human consumption. While people make use of Web information, order products by filling out forms, and get in touch with other people and so on, its usefulness is limited by the fact that information can not be easily understood and processed by machines. The Semantic Web is a vision that makes the Web machine-readable, allowing computers to integrate information from disparate sources to achieve the goals of end-users [4].

2.2 The Semantic Web

The Semantic Web\(^ {19} \) has been proposed by Tim Berners-Lee, the director of the World Wide Web consortium, as a vision to augment the syntactic Web so that the resources are given well-defined, content-related and mutually-agreed meaning. The aim of the Semantic Web is to realize the full potential of the Web by enabling software agents (intelligent software on the Web) to understand, process and aggregate information autonomously and collaboratively.

The realization of the Semantic Web depends on the ability to semantically markup resources through annotations which are machine-understandable. On-

\(^ {19} \)http://www.sciam.com/article.cfm?id=the-semantic-web
Ontologies have been proposed to provide precise and non-ambiguous meaning to Web resources, enabling software agents to understand them. Ontologies are the specification of conceptualizations [28]. Ontology languages are built on the technologies such as XML, Unicode\textsuperscript{20} and Uniform Resource Identifier (URI)\textsuperscript{21} by a layered architecture, which is often represented using a diagram first proposed by Tim Berners-Lee, with many variations. Figure 2.4 gives a typical representation of this diagram.

![Figure 2.4: The architecture of Semantic Web](image)

Unicode is the standard for computer character representation. It is a character code that provides a baseline for representing characters used in most of the languages in the world. URIs provide a mechanism for identifying and locating available resources (such as pages) on the Web.

\textsuperscript{20}http://unicode.org/

\textsuperscript{21}http://www.w3.org/TR/uri-clarification/
Extensible Mark-up Language (XML) is a universal meta language for defining markup. It uses metadata, i.e. data about data, to define tags of their own. These user-defined tags add information to data. This ability permits greater flexibility. However, some mechanisms must exist for coordinating the meaning of the user-defined tags and for understanding the context of that information. XML does not provide any means of talking about the semantics (meaning) of data.

Semantic Web ontologies give precise and non-ambiguous meaning to Web resources, enabling software agents to “understand” them. An ontology is a description of the concepts and relationships for a particular application domain. Ontology languages are the building blocks of the Semantic Web. The Semantic Web languages are constructed on top of XML, Unicode and Uniform Resource Identifier. In the following section, some important languages of the Semantic Web are briefly presented.

2.2.1 Languages in the Semantic Web

The first layer of the Semantic Web is provided by the Resource Description Framework\(^2\) (RDF). RDF is a simple metadata representation framework that uses URIs to identify Web-based resources and graph models for describing relationships between resources. XML describes documents, where RDF describes actual things. It is a model of metadata. RDF provides a simple subject-predicate-object triple structure to make statements about Web resources. Subject is the resource we are interested in, predicate specifies the property or characteristic of the subject and object states the value of the property. The graph corresponding to the statements is shown as follows. An example of a statement is: Prof. Christel Baier is the lecturer of the course Advanced Logics.

\[
\begin{array}{|c|c|c|}
\hline
\text{Course Title} & \text{Lecturer} & \text{Scientific Assistant} \\
\hline
\text{Advanced Logics} & \text{Prof. Christel Baier} & \text{Walter Nauber} \\
\text{Complexity and Logic} & \text{Prof. Franz Baader} & \text{PD Dr. Carsten Lutz} \\
\hline
\end{array}
\]

Table 1: Available Courses

\(^2\)RDF Primer: http://www.w3.org/TR/REC-rdf-syntax/
The representation of statements is shown in Figure 2.5:

```xml
<?xml version='1.0'?>
<rdf:RDF
   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
   xmlns:cd='http://www.description.org/cd#'>
   <rdf:Description rdf:about='http://www.description.org/cd/AdvancedLogics'>
     <cd:lecturer>Prof.Christel Baier</cd:lecturer>
     <cd:assistant>Walter Nauber</cd:assistant>
   </rdf:Description>

   <rdf:Description rdf:about='http://www.description.org/cd/ComplexityandLogic'>
     <cd:lecturer>Prof.Franz Baader</cd:lecturer>
     <cd:assistant>PD Dr.Carsten Lutz</cd:assistant>
   </rdf:Description>
</rdf:RDF>
```

Figure 2.5: RDF document

RDF Schema\(^{23}\) (RDFS) is a simple type modeling language for describing classes and properties of resources in the RDF model. It provides a simple reasoning framework for inferring types of resources.

RDF and RDF Schema provide the core vocabularies and structures to describe Web resources. However, there are a number of disadvantages for RDF Schema. For instance, the agent understood some Web resources ambiguously, namely these resources are not strictly structured. RDF Schema is quite primitive as a modeling language for the Web. Also, it offers certain modeling primitives with fixed meaning, but many desirable modeling primitives for users are missing.

\(^{23}\)Resource Description Framework (RDF) Schema specification: http://www.w3.org/TR/rdf-schema/
Based on RDF Schema, the DARPA (Defense Advanced Research Project Agency) Agent Markup Language (DAML) has been built with a much richer set of language constructs to express classes and properties relationships and more refined support for data types. DAML is the American standard, and has been developed under the lead of DARPA on behalf of the “Department Of Defense” (DOD). DAML combined efforts with the European standard “Ontology Interface Layer” (OIL) and it is now referred to as DAML+OIL. The advantage of DAML+OIL is the ability of defining new classes and properties by defining restrictions on existing classes and properties.

The Web Ontology Language (OWL) was a new ontology language. It was published by the W3C to replace DAML+OIL. OWL was drafted in 2002 and became a W3C recommendation in 2004. OWL provides more vocabularies for describing resources and more complex constraints on their properties. OWL is a suite of language consisting of three species: OWL Lite, OWL DL and OWL Full, with increasing expressiveness. OWL Lite is the least expressive sublanguage. It is obtained by restrictions on the usage of OWL Full language constructs. OWL Lite is easier to grasp for users and easier to implement for tool builders, but the expressivity was more restricted. For example, in OWL Lite, cardinality constraints can only be 0 or 1. OWL Full is fully compatible with RDF, both syntactically and semantically. It is unrestricted full OWL vocabulary. The disadvantage of OWL Full is that the language is undecidable. OWL DL is more expressive than OWL Lite but it is also a subset of OWL Full.

Although we have taken into consideration the different needs of various user groups when designed OWL. OWL is still not powerful enough. OWL provides rich language constructs for describing classes relationships, but it does not provide enough language primitives for describing properties. In the light of this, Logic and Proof are provided. Logic and Proof are reasoning systems that are provided on top of the ontology structure to make new inferences. Reasoning support is usually provided by mapping an ontology language to a known logical formalism.

\[24\text{http://www.daml.org/2001/03/reference.html}\]
\[25\text{http://www.w3.org/2004/OWL/}\]
and using automated reasoners for these formalisms. Description Logics (DLs) are a sublanguages of first-order logic and provide efficient reasoning support for the Semantic Web. The details will be presented in subsection 2.2.2.

The new language, OWL Rules Language (ORL), a rules extension to OWL, was proposed by Horrocks and Patel-Schneider in 2004. They proposed a rule extension (Horn clause rules) to OWL DL. ORL is now known as SWRL\textsuperscript{26} (Semantic Web Rule Language), with some sets of built-ins for handling data type, such as numbers, booleans, string, etc [35]. The one of the main motivations of the rules extension is to infer knowledge not present in the ontology. Although SWRL extends the expressiveness of OWL, it is still limited in expressing certain properties.

The Web Rule Language\textsuperscript{27} (WRL) is designed to be complementary to OWL. It is based on deductive database and logic programming. The WRL is a proposal of rule-based ontology language [19]. At present, most of the research has focused on the rule and reasoning layer of the Semantic Web. This layer is still an active area.

The proof involves the deductions, interchange of proofs, and proof validation. Proof Markup Language (PML) helps users encode and share their proofs in RDF.

The final layer that the Semantic Web can support, is the trust layer. If there are evidences from other trusted internet sources, the claims can be verified. However, this component currently is lack of more significant progress.

2.2.2 Description Logics and the Semantic Web

DL-based Web ontologies play a key role in the Semantic Web. Description Logics [22] (DLs) are a family of logic-based knowledge representation languages that can be used to represent and reason about the knowledge of an application domain in a structured and formally well-understood way. They are a subset of first-order logic (FOL) by selecting certain features to include. The basic notions of DLs are

\textsuperscript{26}http://www.w3.org/Submission/SWRL/
\textsuperscript{27}http://www.w3.org/Submission/WRL/
Chapter 2. State-of-the-art

concepts and roles. Intuitively, a concept represents a set of objects sharing some common characteristics, while a role represents a binary relationship between objects, or between objects and data values.

DLs are made decidable by limiting their expressivity, so that core reasoning services, namely subsumption, satisfiability and instantiation can be solved in full automation. DLs are equipped with a model theoretic, logic-based semantics. Knowledge bases (KBs) can be set up by a knowledge representation (KR) system which is based on DLs. A DL knowledge base is composed of two components: the TBox introduces the terminology, i.e., the vocabulary of an application domain, and the ABox contains the assertions about named individuals in terms of this vocabulary. KR systems also provide facilities to do reasoning about the contents of knowledge bases.

DLs emphasize on the reasoning problems. We can deduce implicitly represented knowledge from the knowledge that is explicitly contained in the KB by decidable reasoning services, such as tableaux algorithms. There is a core inference problem for concepts and individuals of DLs, namely classification. Classification of concepts is also called concepts subsumption which determines subconcept/superconcept relationships between concepts by that we can build a hierarchy structure to provide information on the connection between concepts. Classification of individuals checks whether a given individual is an instance of a concept in terms of the description of the individual and the definition of the concept. DLs provide the basis for the Web Ontology Language (OWL). A concept in DL is referred to as a class in OWL; a role in DL is referred to as a property in OWL.

In modern DLs, the language $\mathcal{ALC}$ (stands for Attributive Language with Complement) [22] is a significant and expressive representative of the various DLs, and it is extended from $\mathcal{AL}$ (a minimal language that is of practical interest) which is introduced in [51]. $\mathcal{ALC}_{R+}$ is an extension of the classical DL $\mathcal{ALC}$ with transitive role axioms($R^+$). In order to avoid names becoming too cumbersome when adding letters to represent additional features for $\mathcal{ALC}_{R+}$, we call this language $S$ [40].
2.2. The Semantic Web

OWL 1.1\(^{28}\) extends the W3C OWL with a small but useful set of features, such as extra syntactic sugar, new Description Logic constructs and expanded datatype support, etc. It provided extra Description Logic expressive power. OWL 1.1 was proposed as a result of the workshop in December 2006. As we have known, \(SH, SHI\) [38], \(SHIQ\) [36], \(SHIO\), and \(SHOIN\), etc. are included in \(S\)-family languages. Well known DAML+OIL is equivalent to the \(SHOIQ\) (\(D\)), OWL Lite and OWL DL is respectively equivalent to \(SHIF\) (\(D\)) and \(SHOIN\) (\(D\)). According to the following three expressions we could summary the features of \(SHOIN\) (\(D\)), \(SROIQ(D^+)\) and \(SHOIQ(D)\):

**OWL DL:** \(SHOIN(D)\) with \((n \geq 0)\) concrete domains \(D\) and GCIs (general concept inclusions)

\[
\{ A, \top, c, c^+, C \sqcap D, C \sqcup D, \neg C, \exists_r C, \forall_r C, r \sqsubseteq s, \{i_1, \ldots, i_n\}, r^{-1} \geq nr, \leq nr \}
\]

**OWL 1.1:** \(SROIQ(D^+)\) provides extra DL expressive power, moving from \(SHOIN\) that underlies OWL DL to the \(SROIQ\) DL. The additions are: qualified cardinality restrictions \(Q\), local reflexivity restrictions for simple properties; reflexive, irreflexive, symmetric, and asymmetric properties; disjoint properties; and property chain inclusion axioms

\[
\{ A, \top, c, c^+, C \sqcap D, C \sqcup D, \neg C, \exists_r C, \forall_r C, r \sqsubseteq s, s \circ s, U, \exists r \text{Self}, \{i_1, \ldots, i_n\}, r^{-1} \geq nr, \leq nr \}
\]

**SHOIQ(D):** \(SHOIQ\) is obtained from \(SHOIN\) by allowing, additionally, qualified number restrictions, i.e., concepts of the form \((\geq nrC)\) and \((\leq nrC)\), where \(r\) is a simple role and \(n\) is a non-negative integer. \(SHOIN\) is the restriction of \(SHOIQ\) where qualifying number restrictions may be only be of the form \((\geq nr\top)\) and \((\leq nr\top)\). In this case, we omit the symbol \(\top\) and write \((\geq nr)\) and \((\leq nr)\) instead.

\[
\{ A, \top, c, c^+, C \sqcap D, C \sqcup D, \neg C, \exists_r C, \forall_r C, r \sqsubseteq s, \{i_1, \ldots, i_n\}, r^{-1} \geq nrC, \leq nrC \}
\]

The DL \(\mathcal{EL}\) allows conjunction and existential restrictions than its counterpart \(\mathcal{FL}_0\), which allows conjunction and value restrictions [21]. The DL \(\mathcal{EL}^+\) is ex-

\(^{28}\)http://www.w3.org/Submission/owl11-overview/
tended by $\mathcal{EL}$ with so-called role inclusions (RI) [23]. It is well-suited for formulation of medical ontologies, such as Galen Medical Knowledge Base (GALEN) [49], and biological ontologies, such as Gene Ontology (GO) [20].

### 2.2.3 Tools in the Semantic Web

Besides ontology languages, a large number of development tools, including application, creation, management, querying and verification, etc., have emerged to support the Semantic Web in recent years. Here we briefly introduce information of these tools: libraries, reasoners and editors.

Jena\textsuperscript{29} is a Java API for building Semantic Web applications. It was developed at HP Labs at Bristol [26]. Jena is used to hold and output the instances into an RDF file. Hence it is a good writer for RDF.

Reasoners, also called inference engines, are software applications that derive new facts or associations from existing languages. The use of reasoners in the Semantic Web allows applications to inquire why a particular conclusion has been reached, and reasoners are the ontology storing and querying tools.

KAON2\textsuperscript{30} was developed by Information Management Group, University of Manchester. It is available as a pre-compiled binary Java distribution and is free for research and academic purpose. The latest version was released in August 2006. KAON2 is an infrastructure for managing OWL and F-Logic [44] ontologies. Reasoning is implemented by resolution-based algorithm that reduces a description logic knowledge base to a disjunctive Datalog\textsuperscript{31} program. KAON2 supports OWL DL $\mathcal{SHIQ}(\mathcal{D})$. This includes all features of OWL DL except nominals. Reasoning within KAON2 is sound and complete that has been proved [41].

Pellet\textsuperscript{32} was developed by University of Maryland. It is a DL reasoner based

\footnote{\textsuperscript{29}All official release of Jena can be obtained from: http://jena.sourceforge.net}

\footnote{\textsuperscript{30}Please visit http://kaon2.semanticweb.org/ for more details about how to use KAON2.}

\footnote{\textsuperscript{31}http://en.wikipedia.org/wiki/Datalog}

\footnote{\textsuperscript{32}http://www.mindswap.org/2003/pellet/}
on well-known tableau algorithm, and deals with TBox (Terminology Box) reasoning and non-empty ABox (Assertion Box) reasoning [52]. Pellet is completely written in Java. At present, the latest version 1.5 is available [5]. Pellet is sound and complete by incorporating the decision procedure for DL \( SHOIQ \) [37] (the expressivity of OWL DL plus qualified cardinality restrictions in DL terminology).

**Fast Classification of Terminologies (FaCT)** is a TBox reasoner that supports automated concept level reasoning, namely class subsumption and consistency reasoning. FaCT was developed by University of Manchester. It implements a reasoner for the description logic \( SHIQ \). Given an OWL ontology, FaCT can classify the ontology to reduce redundancy and detect any inconsistency within it. It does not support ABox reasoning. FaCT++\(^{33}\) is the descendent of FaCT [34] reasoner. It provides full support for OWL Lite. The current available version 1.1.11 was released on 28 March 2008. It supports OWL DL reasoning as well as the forthcoming standard OWL 1.1 (\( SROIQ(D^+) \)).

Racer\(^{34}\) (Renamed ABox and Concept Expression Reasoner) was developed by Hamburg University of Science and Technology Software, Technology, and Systems (STS). It implements a highly optimized tableau calculus supporting multiple TBox and ABox reasoning for the description logic \( SHIQ \) [31]. The logic \( SHIQ \) [40] is the extension of the logic \( ALCN^\mathcal{H}_{R^+}(D) \) [30] by additionally providing qualified number restrictions and inverse roles. Race (Reasoner for ABoxes and Concept Expressions) supports the logic \( ALCN^\mathcal{H}_{R^+} \), and Race is the precursor of Racer. The Racer system has been commercialized and RacerPro is the new name of the software. It appeared in 2002 and is a lisp-based reasoner. The current release 1.9.2 is the latest version from Racer System\(^{35}\). RacerPro has recently been continuously improved and supports the full OWL standard. RacerPro is available for Linux, Windows, and MacOS X. The Semantic Web reasoning tools are used for the knowledge base classification and concept subsumption. Racer provides reasoning support for instances, consistency checking and the subsump-

\(^{33}\)http://owl.man.ac.uk/factplusplus/

\(^{34}\)http://www.sts.tu-harburg.de/r.f.moeller/racer/

\(^{35}\)http://www.racer-system.com/
CEL\textsuperscript{36} (A polynomial-time classifier for the description logic $\mathcal{EL}^+$) is the first reasoner for the description logic $\mathcal{EL}^+$, which can be used to compute the subsumption hierarchy induced by $\mathcal{EL}^+$ ontologies. It was developed by University of Dresden. The latest version CEL 1.0beta has been released in January 2008.

Ontology editors can help ontology engineers focus on the semantics without worrying much about syntactic organization. A good ontology editor can save large amount of time when developing ontologies. The editor most frequently cited was Protégé\textsuperscript{37}. It is a free, open-source, Java-based Semantic Web ontology editor/creator and knowledge-based framework, and it provides an extensible architecture, allowing users to create customized applications. Protégé was developed by Stanford University for developing knowledge-based systems. Protégé 3.4 beta is the latest version. One can download it from [6]. Protégé could assist in ontology evolution, and could help users to prevent or circumnavigate common design mistakes.

OilEd\textsuperscript{38} is a graphical ontology editor that has an easy to use frame interface, supports the construction of OIL-based ontologies. It was developed by the University of Manchester. OilEd offers a frame paradigm to deal with a very expressive language DAML+OIL. The key aspect of OilEd’s behaviour is the use of the FaCT reasoner\textsuperscript{39} to classify ontologies and check consistency via a translation from DAML+OIL to the $\mathcal{SHIQ}$ description logic\textsuperscript{42}. However, OilEd lacks many features that would be required of a fully-fledged ontology development environment, e.g. it does not support for working with multiple ontologies.

Protégé and OilEd both support reasoners that conform to the DIG\textsuperscript{39} (Description Logic Implementation Group) interface. DIG interface provides Java API to communicate with DIG compliant reasoners, such as Racer and FaCT++. The DIG

\textsuperscript{36}CEL can be downloaded from http://lat.inf.tu-dresden.de/systems/cel/
\textsuperscript{37}http://protege.stanford.edu/
\textsuperscript{38}http://oiled.man.ac.uk/
\textsuperscript{39}http://dl.kr.org/dig
2.2. The Semantic Web

protocol is an XML- and HTTP-based standard for connecting clients (applications) to DL inference engines through the use of HTTP POST request. The following example shows how to use the TELL request from clients and a reasoner. DIG allows for creating or releasing KBs and enables clients to pose standard DL queries. The main ideas of DIG are described in detail in [24].

Example 1. *The TELL statements define three classes respectively, Human, Man, and YoungerMan in following Figure 2.6,*

```xml
<tells>
  <releaseKB/>
  <defconcept name = 'Human'/>
  <defconcept name = 'Man'/>
  <equalc>
    <catom name = 'youngerMan'/>
    <and>
      <catom name = 'Man'/>
    </and>
    <intmin val = '18'>
      <attribute name = 'age'/>
    </intmin>
  </equalc>
  <defattribute name = 'age'/>
</tells>
```

Figure 2.6: Sample TELLs

If the ontology is loaded into the knowledge base (kb), for example, we can use the reasoners API in following:

```java
   tells.getTells().setUri(kb);
   ResponseDocument response = reasoner.tells(tells);
```

SMORE\textsuperscript{40} is shorthand for the Semantic Markup, Ontology, and RDF Editor.

\textsuperscript{40}http://www.mindswap.org/2005/SMORE/
It is a semantic annotation tool that provides users with an integrated environment for creating Web pages. SMORE is associated with user-specific terms and elements in order to allow users to markup their documents in RDF using Web ontologies. Users can also create own ontologies from existing ontologies, or modify and extend the existing ontologies. The release named SMORE 5.0 is the latest version. It can be downloaded from [7].

2.2.4 Current Research on the Semantic Web

There are a large number of researchers working in the area of the Semantic Web both in academia and in industry, for example active research groups in HP Laboratories, in Knowledge System and AI Laboratory of Stanford University, or in Automata Theory Group of TU Dresden. These researchers are playing a leading role in all aspects of Semantic Web development. Many projects are also sponsored by the European Commission, as well as to the rest of world. For example, at present, most of researchers are exploring how to combine the distributed nature of the Web with the power of semantic description, logic and reasoning. However, the most widely developed space at the moment within the Semantic Web is the organisation and discovery of Web information. It is the primary motivation behind Semantic Web’s development. Knowledge discovery and knowledge management are many examples of application researchings, that they could dramatically change information through the introduction of Semantic Web technology.

The applications of the Semantic Web are currently limited. Applications have not largely caught on to exploiting the social mechanisms that are powering the Web 2.0 sites. A more elaborate scenario for the Semantic Web involves not only the facility for improved searching, but also the facility for selecting, assembling and triggering services found on the Web. Researchers have focused on trying the missed point that was precisely the space where the semantics are needed and must easily be exploited. Therefore, exploiting many Semantic Web applications that focus solely on expert system applications with expressive semantics and Web nature of the Semantic Web become a research direction.
Of course, there also are some important researchs in other fields. These include document management for manipulating complex markups, interface design for offering new views of these marked documents, and agent for gathering knowledge and negotiating over the network. E.g., the research project “Acquisition for Networks of Negotiating Agents” in Intelligence, Agents, Multimedia Group of Southampton University aims to investigate and develop techniques by which software agents can acquire sufficient knowledge to negotiate effectively on behalf of their user in a range of electronic commerce scenarios [8].

The Semantic Web crucially relies on the possibility to integrate multiple ontologies. This is known as the problem of ontology mapping or ontology integration, and is indeed one of the most active areas of research in the Semantic Web community. A wide array of techniques is deployed for solving this problem with ontology mapping techniques based on natural language technology, on machine-learning, on theorem-proving, on graph-theory, on statistics, etc. Although encouraging results are obtained, this problem is by no means solved, and automatically obtained results are not yet good enough in terms of recall and precision to drive many of the intended Semantic Web use-cases. E.g., the research project “Modularity of Ontologies” in Information Management Group of Manchester University aims to develop meaningful notions and related reasoning services for composing and decomposing ontologies to serve as the basis for the collaborative development and re-use of ontologies [9].

The Semantic Web has sometimes been criticized as being too much about “Semantic”, and not enough about “Web”. This was perhaps true in the early days of Semantic Web developments, where there was focus on applications rather in circumscribed domains like intranets. Recent years a large number of developments in the Web-aspects of Semantic Web technology have been seen. A prime example of this is not just textual documents, non-textual media such as images and videos, special multimeida become also a part of the Web. For example, a multimedia system based on Semantic Web technology becomes one hot point in research areas. E.g., the research project “Multimedia Thesaurus and Intelligent
Agent support for content based retrieval” in Department of Electronics and Computer Science of Southampton University aims to provide a programme of research to develop multimedia architectures for video, image and sound [10].

Except as above mentioned aspects, other interesting research projects also attract significant attention in the Semantic Web research community. For example, automated deduction are concerned by Automata Theory Group of Institute for Theoretical Computer Science at TU Dresden. They are mainly interested in sub-classes of first-order predicate logic for which the interesting inference problems are still decidable. They are researching from two sides; on one side, they are investigating the combination of special deduction methods, and their integration into general deduction procedure. On the other side, they are designing logic-based knowledge representation languages with decidable inference problems. Current research projects in this group are:

• Completing Knowledge Bases using Formal Concept Analysis
  As far as we know the most notable success of DLs is that it provides the logical underpinning of OWL, the standard ontology language for the Semantic Web. Due to the standardization by the W3C, more and more users wrote ontologies in OWL for applications, the size of ontologies become huge, tools availability for improving their quality becomes more important. However, until now we only can use DL reasoning to detect inconsistencies and to infer consequences from implicit knowledge that can be deduced from the explicitly represented knowledge. These approaches depended on the soundness of an ontology. In this project they aim to develop formally well-founded techniques and tools that depended on the completeness of ontology. User using this can check whether ontology contains all the relevant information about the application domain, and to extend the ontology appropriately if this is not the case [11].

• Explaining Ontology Consequences
  The aim of this research project is to develop methods for supporting knowledge engineers in diagnosing and correcting errors in the built ontologies. The approach is to find small sub-ontologies from which a given consequence follows, these sub-
ontologies are called explanations, and modify the procedures used to detect the consequence to allow for tracking the ontology axioms responsible for it [12].

- Description Logics with Existential Quantifiers and Polynomial Subsumption Problem and Their Applications in Bio-Medical Ontologies
In this project they concentrate on investigating description logics with existential restrictions both theoretically and practically. It includes identifying the polynomial borders of subsumption problems, developing high optimizations for the subsumption algorithms, evaluating the application in large-scale bio-medical ontologies. Moreover, reasoning problems researching, e.g., conjunctive queries, is also considered to be supplemental [13].

Knowledge Systems and AI Laboratory at Stanford University conducts research in the areas of knowledge representation and automated reasoning for the Semantic Web. Current and recent projects focus on enabling technology for the Semantic Web, hybrid reasoning, and knowledge-based technology for intelligence analysts. For example, DARPA Agent Markup Language (DAML) Research Project, Cognitive Agent that Learns and Observes (CALO) Project [14].

The goal of DARPA Agent Markup Language (DAML) Research Project is to develop next generation Semantic Web tools and technologies for DAML, e.g. DAML Language Research, DAML-Enabled Web Services, DAML Tools Research and so on [15]. The research of Cognitive Agent that Learns and Observes (CALO) Project focuses on object-oriented modular automated reasoning [16].
Chapter 3

Motivation for Participating in the Semantic Web

This chapter presents the motivation for participating in the Semantic Web. It is divided into two parts. In first part, we give a brief general motivation description. Following that, three examples of application scenarios from the different domains are illustrated in the second part.

3.1 General Aspects

The World Wide Web today is a huge network of information resources. It has dramatically changed the availability of information. While information resources were published in the Web, they were not suitable for software agents. They are only designed to be suitable for interpretation by Web browser.

At the same time, the Web currently contains around 3 billion static documents, which are doubles in size every six months on the Web. These documents are accessed by over 300 million international users [48]. This enormous amount of data has made it increasingly difficult to search, filter, convert, interpret and summarize the information for the benefit of the user. For example, the Web search engines search only few of potentially relevant sites and return a lot of unwanted information. Another side, the current Web remains largely unstructured, and
large amounts of information remain unavailable.

Thus, the Semantic Web as an extended Web has emerged. The idea of a Semantic Web is: having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and re-use of data across various applications. Hence it represents a set of technologies that will work well on internal corporate intranets. Uses of the Semantic Web improve e-business processes and business-to-business communication. To sum up, in the Semantic Web, several key problems facing current information technology architecture will be resolved, such as information overload, information retrieval, knowledge management and content aggregation, which include not only static data, but also dynamic Web services.

### 3.2 Motivating Scenarios

Several real-life applications of ontologies from the different domains are introduced in this section, and we use these motivating scenarios to illustrate why ontologies are needed.

#### 3.2.1 Intelligent Search Scenario

We present a Semantic Web scenario from tourism domain, an online traveling booking system. Figure 3.1 shows the ontologies of an travel booking system. Its basic idea is that providers of travel-related services advertise their services on the Semantic Web, so that traveling booking agents can find these services such as transportation, accommodation and entertainment dynamically. The traveling agent provides automated booking services to users.

At the beginning, a user submits a request, which includes the information about traveling time, preferred areas, frequent flyer airlines, preferred entertainment and maximum budget to the traveling booking agent. According to the received request, the traveling booking agent could give suggestions on vacation planning and is able to plan a complex travel arrangement to meet the user’s pref-
3.2. Motivating Scenarios

If the total cost overruns the budget limit, the traveling booking agent tries to find another cheaper arrangement. If there is no combination that can be found within the budget, the traveling booking agent will notify the user. Otherwise it shows the most suitable travel arrangement to the user based on information from the user’s preferences. If the user is satisfied, he/she submits his/her credit card information to the traveling booking agent. The traveling booking agent asks a credit checking agent to check if the card is valid. If it is, the booking will be made. This is a simple scenario, which the agent can split into several simple semantic services such as entertainment, payment, accommodation, and arrange-transport and so on.

Figure 3.1: Travel booking system
3.2.2 Medical Semantic Scenario

With the rapid development of the healthcare industry, there is a large amount of data describing procedures, treatments, diagnoses, drugs and insurance plans about patient. Medical practitioners face the challenge in managing their records and spend more time to report each new patient encounter which contain a problem list, family history, history of present illness and review of symptoms.

The Semantic Medical Record application uses Semantic Web technologies to reduce medical errors, improve physician efficiency and manage all patient records. The physicians can easily capture and analyze health outcome that is provided by a system which has rich domain through the use of ontologies and rules. Documents get semantic feature by automatic semantic annotation of documents with respect to one or more ontologies.

The most important benefit with the semantic annotation for medical areas is the reduction of medical errors that occur as an oversight. For example, a given drug ontology may contain all the drugs and classes of drugs and drug allergies. Capturing such informations are important to not allow a patient to be prescribed two severely interacting drugs and prevent the drug interaction. These reduce medical errors and increase patient safety. The medical semantic application expedites and enhances the patient documentation process, especially in small clinics. Figure 3.2 shows the partial view of drug ontologies.

3.2.3 An E-learning Scenario

Traditional learning is driven by the instructor. The instructor selects the content, and sets agenda and pace of learning. Learning takes place at specific time and specific places.

E-learning is an area which can benefit from Semantic Web technologies. The Semantic Web could offer more flexibility in e-learning systems through the use of such technologies as annotation tools and integration. Depending on student interests and needs, knowledge can be accessed in any order the student wishes.
3.2. Motivating Scenarios

Figure 3.2: Partial drug ontologies

The Semantic Web can provide an uniform platform for the business processes of organizations, and learning activities can be integrated into these processes.

Imagine some one is studying complexity in Description Logic. The teacher has not provided the relevant link to the concept, so he enters “Complexity” in the search form. The result list shows that Complexity occurs in several contexts, and he decides to have a look at Complexity in an approximation context, which seems most appropriate for his current studies.

After having looked at the different kinds of contexts, he wants to see if there are any appropriate learning online courses. He draws a query for “description logic resource in English related to complexity that is on the university course at Artificial Intelligence Institute”. The agent handles the request and returns a set of choices satisfying the query. If no course is found, he can register with a notification service. Otherwise, he may find a suitable course and then make a final decision about registering for the course.

Processing the registration can be seen as a complex service involving registering with resource management, creating a confirmation notification, creating account for him, and providing learning materials. Once all these are done, he can login and start the course.
Chapter 3. Motivation for Participating in the Semantic Web
Chapter 4

Implementation Details

In this chapter, we demonstrate the implementation of transforming a general Web application to the Semantic Web. At first we create a common OWL DL ontology and use it in a simple Web application. The decision regarding applied implementation of libraries and tools, such as Jena, RACER, will be briefly explained.

4.1 Transforming a Web Application to the Semantic Web

4.1.1 Ontology Creation

We need to create a common OWL DL ontology by using an ontology editor, such as Protégé, and include it into a simple Web application. We choose OWL DL due to its correspondence with a well studied description logics (hence the suffix DL). Description Logic is a decidable\(^1\) fragment of first-order logic and therefore is amenable to automated reasoning [46]. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for efficient reasoning systems. Hence, it is able to automatically compute the classification hierarchy (also known as subsumption reasoning) and check for inconsistencies in an ontology.

\(^1\)Logics are decidable based on the logic will terminate in a finite time.
In order to deal with practical applications, we need to install an appropriate software environment. For this thesis the decision has been made to use: Tomcat 5.5\textsuperscript{2}, MySQL\textsuperscript{3} and Ant\textsuperscript{4} due to their advanced implementation and easy usability.

This application simulates an article finding system of a news portal which makes it possible to store articles with their relevant title, author, and content and so on, in a database. The stored information is available for customers via a control RDF file. The RDF annotations are shown on the automatically generated homepages of the articles respectively. To handle this application it is required to create OWL classes, such as Article, OWL individuals, and OWL properties, and to define the scope and domain of properties. OWL properties represent relationships between two individuals. There are two main types of properties, Object properties and Datatype properties. Object properties link an individual to an individual. Datatype properties link an individual to XML Schema Datatype value\textsuperscript{5} or RDF Literal\textsuperscript{6}. For our application, we have only one Article class. Therefore, we do not exist object properties. Properties may have a domain and a range specified. Properties link individuals from the domain to individual from the range.

One of the key features of ontologies that are described using OWL DL is that they can be processed by a reasoner. Reasoners offer one of the main services which is to test whether or not one class is a sub-class of another class (also known as subsumption\textsuperscript{7} test). It is possible for a reasoner to infer ontology class hierarchies by performing such tests on all of the classes in an ontology. Reasoners offer another standard service which is consistency checking. Based on the description of a class, the reasoner can check whether or not it is possible for the class to have any instances. A class is believed to be inconsistent if it cannot have any instances. Ontological consistency can be checked by reasoning engine such as RACER. It is presented in details in the following subsection.

\textsuperscript{2}http://tomcat.apache.org/
\textsuperscript{3}http://dev.mysql.com/downloads/
\textsuperscript{4}http://ant.apache.org/bindownload.cgi
\textsuperscript{5}see http://www.w3.org/TR/xmlschema-2/ for more information on XML Schema Datatypes
\textsuperscript{6}http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal
\textsuperscript{7}the description of the classes (concepts) are used to determine if a super-class/sub-class relationship exists between them
4.1. Transforming a Web Application to the Semantic Web

4.1.2 Decision Regarding Tools

The Semantic Web is often described as the next “big” technology leap for the Internet. Therefore, efficient development tools are a prerequisite for the wide adoption of Semantic Web technology. Our goal is to introduce tools with support for the most commonly needed features for our simple Semantic Web application.

We know that ontologies provide formal models of domain knowledge that can be exploited by intelligent agents. Therefore ontology plays a central role in the Semantic Web. Furthermore, since ontologies may be hard to build manually, some dedicated tools like Protégé or with API like Jena will provide “intelligent” assistance in ontology construction and evolution.

Protégé is one of the most popular tools for editing ontologies. It created the practice for building Semantic Web contents by the needs of and feedback from users. In our application, we use Protégé to create one OWL DL ontology. As illustrated in Figure 4.1, it shows the logical definition of the properties of the class Article.
The major benefit of using Protégé in our application is its open architecture. Users do not need to spend time developing the base infrastructure with standard features such as loading and managing OWL ontologies. Instead, they can use the existing base components as templates for customized solutions. Furthermore, external components like reasoners and Web services can be integrated easily into Protégé.

Jena is a Java framework for building the Semantic Web. It provides a programmatic environment for RDF and OWL. One statistics indicated in [25] that most groups are using Jena. In our application, we use Jena to read and convert an OWL file into a corresponding Java representation, which is a Jena static OntModel. Later, the application will use this static snapshot model to build a specific model based on the content of Web page. This model has to be rebuilt after each change of the Web page.

The API of Racer is not good at manipulating ontologies, but we use Racer to test the consistency and the subsumption problem of OWL DL ontologies. Racer supplies a client which offers a lean interface to communicate with RacerPro via HTTP [27] through the DIG protocol. Protégé can be seen as a specific client that uses the DIG protocol [24] for communicating with a Racer server. Racer provides a socket based interface which can be used by client applications. To access RacerPro from our own application we can use these APIs to the native command set of RacerPro. The libraries utilize the TCP/IP interface to access RacerPro. (The Racer TCP interface can be very easily accessed from Java or C++ application programs as well. For these languages corresponding APIs are available.)

Racer is still unique in its highly optimized tableau-based reasoning support for TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering). Racer can explicitly deal with individuals (nominals) that differs from other reasoners, which provide only concept consistency (TBox) reasoning. Another distinguishing feature of Racer also includes optimization techniques supporting the classification of very large knowledge bases ($\mathcal{T}, \mathcal{A}$). Racer
can be considered as one of the fastest OWL DL reasoners based on sound and complete algorithms that is currently freely available [32]. Furthermore, Racer also supports concrete domain reasoning over constraint-based data types.

### 4.1.3 Realization based on Semantic Annotation

We map an OWL ontology into the Java language by automating the generation of ontology specific class libraries for traditional object-oriented programming languages. An ontology is compiled to produce an ontology specific library. This library is linked with other libraries and codes to produce an application. One approach to building Semantic Web applications is to rely on class libraries for mainstream object-oriented languages (Java). The class library provides classes and fields that hold both ontologies and models built from them. Table 2 exhibits the use of a library to write a fragment of the sort of code that would be used in our ontology application. The code uses the Jena [17] class library for Java. Jena is a library that can manipulate pure RDF models as well as models that use the DAML+OIL [18] extensions.

First Jena reads the Ontology.owl file and converts this model to a Java class, called Ontology.java. The Java file includes a static model which is equal to the model from the input Ontology.owl file. Later, OwlGenerator.java will use the static model and the Jena library to create a dynamic model according to different inputs which come from the static web page. After this, all known properties are added to individuals.

In our application, there are two web pages that can be translated to semantic models. These are article summary web page and article display web page. As these two pages display different information, even though they use the same ontology model, we also must build two different functions within one generator to translate these two web pages to semantic models. On article summary web page, it display every article’s title, author, date and abstract only, so to build the semantic model, we just read all articles’ summary to a list. With this list as input, OwlGenerator.java will build the correspondent semantic model. On
article display web page, every time, there is only one article displayed, so with this specific article as input, OwlGenerator.java will build this web page’s semantic model.

<table>
<thead>
<tr>
<th>Jena API</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.hp.hp1.jena.ontology.Individual</td>
<td>Interface that encapsulates an individual in an ontology, sometimes referred to as a fact, assertion or a member of the ABox.</td>
</tr>
<tr>
<td>com.hp.hp1.jena.ontology.OntModel</td>
<td>An enhanced view of a Jena model that is known to contain ontology data, under a given ontology terminology.</td>
</tr>
<tr>
<td>com.hp.hp1.jena.ontology.OntModelSpec</td>
<td>Extends java.lang.Object, encapsulates a description of the components of an ontology model, including the storage scheme, reasoner and language profile.</td>
</tr>
<tr>
<td>com.hp.hp1.jena.rdf.model.Model</td>
<td>A package for creating and manipulating an RDF model.</td>
</tr>
<tr>
<td>com.hp.hp1.jena.rdf.model.ModelFactory</td>
<td>ModelFactory provides methods for creating standard kinds of Model.</td>
</tr>
<tr>
<td>com.hp.hp1.jena.rdf.model.RDFWriter</td>
<td>An interface to RDF serializers.</td>
</tr>
</tbody>
</table>

Table 2: Jena API descriptions

4.2 General Strategy

In this section, we provide guidance for transforming a Web application to a system of semantically annotated information step by step. This includes:

1. Ontology Engineering

   \( S_1 \): **Creating an ontology** Assemble appropriate information resources and expertise, and define the terms in the domain of interest, acquire domain and knowledge. Design the overall conceptual structure of the
domain. This will involve identifying the relationships among the concepts, distinguishing which concepts have instances, and organizing the ontology. Then ontologies can be built using powerful tools (such as Protégé) and saved as OWL files in the OWL DL format.

$S_2$: **Loading the ontology** Starting up the reasoner, after the reasoner has been successfully loaded, load the ontology.

$S_3$: **Checking Consistency** After having read in and preprocessed the ontology, the reasoner is ready to check syntactic, logical and semantic inconsistencies among the ontology elements. Consistency checking may involve automatic classification that defines new concepts based on individual properties and class relationships. Consistency checking of the ontology reveals the inconsistency of the concept.

$S_4$: **Mapping or Integration** As more and more individual ontologies are created, interoperability among different ontologies becomes important when working with more than one ontology. Thus, mapping or integration of ontologies becomes a necessary step for Web application.

$S_5$: **Checking Consistency Again** After ontology mapping or integration, the ontology needs to be checked for consistency again. Checking consistency will take some time depending on the size and complexity of the ontology.

2. Web Application Engineering

Depending on preceding concrete implementation details we demonstrate an example for Web application engineering.

$S_6$: **Creating Web Application** Install servlet container or application server such as Tomcat and Relational Databases such as MySQL, then deploy the Web application over deployment platform Tomcat, and notify Tomcat that we will be using MySQL. The web application will store information values to a table inside MySQL and will return a JSP page that contains all the records in that table. Write and compile the web component code JSP that is used by the Web components, create any static resources that will be used by the Web components.
such as JSP pages. Once a Web application is deployed, the user can access the Web components through the browser.

\( S_7 \): Integration Semantically Information into Web Application Utilize Jena library to read ontology file and transform this model to a Java class. Then OWL Generator will use the static model and Jena library to create dynamic semantic model according to concrete input, these input are linked to hierarchies of classes and properties, some of the links have a fixed semantics.

4.3 Summary

The main contribution of this chapter is transforming one small practical application to a semantically annotated Web application. To handle this application it is easy to create OWL classes, OWL individuals and OWL properties by the intuitive user interface of Protégé. The ontology is then checked by Racer fully automatically. A model of the associated ontology has been generated by Jena libraries. It is a Java representation of ontology. Based on the ontology this model can be instantiated to generate a concrete application representation that has all associated properties. These properties links have a fixed semantics, e.g. subclass, domain, range. Therefore, the semantic information is associated to Web application during this model.

However, from this application, we discover that there exist some problems in realization for transforming a Web application to the Semantic Web. As every web page might has itself special structure, so it may use different ontology models. In order to create one Semantic Web application, one Web application can have many different pages with different structure, so we need to build special semantic model generators for every page. This results in a big problem regarding maintaining the Web application.
Chapter 5

Evaluation

In the previous chapter we created one Semantic Web application, discussed different editors and reasoners. While introducing the approaches, we came across a number of conceptual issues, some of those resulted in a loss of functionality. In other cases, there are even various ontology tools currently available on the market, but following ontologies are expected to rise in size, we will have to think about reasoning systems in more detail. This chapter provides analysis and evaluation of the tools in depth based on practical applications.

5.1 Protégé vs SMORE

Protégé is easy to use in comparison with many other commercial and open-source ontology editors, but it does require a fundamental knowledge of ontologies and the types of objects and relationships defined within them. Without this knowledge, a beginning user would be completely lost within the tool. However there are numerous sites that provide pre-made ontology templates and guided tutorials that allow new users to work through the basic concepts and gain a solid understanding in a fairly short period of time.

SMORE is another ontology editor that allows users to create and edit ontologies using a interface. It has an integrated Web browser component. This additional functionality allows the user to browse the internet within the program
and to create ontologies from the terminology used on a web page. Aside from the Web integration, SMORE is simple to use for an intermediate user and allows simple and flexible editing of an ontology.

Compared to SMORE, Protégé is easy to integrate custom-tailored components to build real-world applications. Protégé is used to edit and create ontologies in various formats, including RDF and OWL script languages (including OWL Full, DL and Lite) or through its Java-based plug-and-play environment. Protégé provides a tabbed view of an ontology, allowing the user to separate classes from subclasses, classes from properties, classes from individuals, etc., and look at all of the characteristics and relationships attributed to each project. Files can be exported to OWL, N-Triple and TURTLE format. Protégé also provides comprehensive mapping between its open Java API and the standard OWL parsing library Jena. However, users feel more cumbersome to use SMORE. The lack of wizards and less user-friendly interface cannot compare to the wealth of features Protégé has, and it is a chore to try and find what users are looking for [50].

5.2 Evaluation of Reasoners

We analysis and evaluate reasoning tools based on different functionalities in this section.

KAON2 supports ontology languages. It implements a resolution-based decision procedure for TBoxes and ABoxes. Generally speaking, KAON2 seems to do better on large ABoxes reasoning tasks than reasoners which implement the tableaux calculus, especially KAON2 is known to perform very well if ABox is large and TBox is of medium size [47]. Following some testing cases with small TBox [45], KAON2 either answers immediately or indicate the system ran out of memory. In particular if cardinality restrictions with cardinalities greater or equal to two KAON2 often runs out of memory. From these practical cases, we can see that KAON2 does better on reasoning with large amounts of individuals, but not good at dealing with TBox services. Although KAON2 manipulates OWL, it can not create and load the ontology with the syntax for hedges (or adverbs). KAON2 provides a stand-alone server or library within Java. In addition it supports the
5.2. Evaluation of Reasoners

Racer supports multiple TBoxes and ABoxes. In addition, Racer also provides support for retraction of assertions in particular ABoxes. The inference services supported by Racer for TBoxes and ABoxes are described in detail in [29]. Racer is able to reason with datatypes of type String, Integer, and Real. Nevertheless, it currently does not support the newly introduced role expressions of $\text{SROIQ}(D^+)$.

From the test case [45] we know that Racer was among the fastest systems when TBox is of medium size. Racer offers flexible interfaces. It supplies a client which offers a lean interface to communicate with RacerPro via HTTP by using the DIG protocol. The current extension of Racer is its new ABox query language nRQL [33] (new Racer Query Language) that can be used to provide access to extensionally specified information in ABoxes.

The latest version FaCT++ not only supports complex restrictions for Integer and String datatype, but also supports a fraction of the role language feature ($\mathcal{R}$) of OWL 1.1\textsuperscript{1}. Test cases [45] show that FaCT++ is implemented extremely stable. FaCT++ can be used either as a standalone DIG-enabled reasoner or as a servlet implementing the HTTP DIG interface. FaCT++ supports the TBox reasoning tasks such as subsumption and consistency checking as well as taxonomy construction. It also implements instance classification. However, FaCT++ is a traditional prover; this architecture limited its reasoning features.

Pellet supports the DL $\text{SHOIQ}(D)$ which corresponds to OWL DL with user-defined datatype [53]. In comparison to other reasoners, Pellet is the only system which claims to support the full XML Schema datatype. Pellet can read OWL syntax and is executed via command line. It is used as the underlying OWL reasoner for SWOOP ontology editor [43]. Detecting unsatisfiable classes, checking for entailed statements and building the class taxonomy are the available inference services of Pellet. Pellet also supports the conjunctive query languages SPARQL\textsuperscript{2}.

\footnotesize{\textsuperscript{1}http://www.w3.org/Submission/owl11-overview/\textsuperscript{2}http://www.w3.org/TR/rdf-sparql-query/}
and RDQL\textsuperscript{3}. In addition the most special difference is that Pellet supports ontology debugging. Pellet can diagnose why the error occurs when detection of unsatisfiable concepts in an ontology. It is the only one currently to support this function.

CEL is based on a polynomial-time subsumption algorithm. This makes CEL distinguishing with other modern DL reasoners. This polynomial-time algorithm allows CEL to process very large ontologies in reasonable time. This theoretical advance that has shown that the description logic $\mathcal{EL}$, which allows for conjunction and existential restrictions, and some of its extensions has a polynomial-time subsumption problem and so-called general concept inclusions (GCIs) [21]. Moreover the CEL provides the command line options and the DIG server. Since version CEL-0.9beta, the DIG interface has been available, now CEL can work as a back-end reasoner for a graphical ontology editor, such as Protégé. Through a number of experimental results [23] it has been shown that CEL can outperform the fastest tableau-based DL systems and CEL is suitable for reasoning on very large life science ontologies.

In order to gain an overview, table 3 summarizes ontology reasoners in Query language, Support language, Inference support, API support, and Efficiency, etc.

\textsuperscript{3}http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
### 5.2. Evaluation of Reasoners

<table>
<thead>
<tr>
<th>Tools</th>
<th>Query Language</th>
<th>Support Language</th>
<th>Inference Support</th>
<th>API Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Racer</td>
<td>nRQL</td>
<td>DL</td>
<td>Especially for ABox retrieval</td>
<td>DIG 1.1</td>
</tr>
<tr>
<td>KAON2</td>
<td>F-logic, OWL</td>
<td>OWL</td>
<td>OWL-DL</td>
<td>Java</td>
</tr>
<tr>
<td>FaCT++</td>
<td>no</td>
<td>DL $SHIQ$</td>
<td>yes</td>
<td>DIG</td>
</tr>
<tr>
<td>Pellet</td>
<td>SPARQL, RDQL</td>
<td>$SHOIQ(D)$</td>
<td>yes</td>
<td>DIG</td>
</tr>
<tr>
<td>CEL</td>
<td></td>
<td>DL $\mathcal{EL}^+$</td>
<td>yes</td>
<td>DIG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th>Update Support</th>
<th>Implementation Language</th>
<th>Export data format</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Racer</td>
<td>yes</td>
<td>Java, C++</td>
<td>Java</td>
<td>Large tableau-based ABoxes in combination with large TBoxes</td>
</tr>
<tr>
<td>KAON2</td>
<td>yes</td>
<td>Java, python</td>
<td>Any(Java)</td>
<td>ABox is large, TBox is less than or equal to mediumsize</td>
</tr>
<tr>
<td>FaCT++</td>
<td>yes</td>
<td>C++</td>
<td>?</td>
<td>Standard TBox reasoning</td>
</tr>
<tr>
<td>Pellet</td>
<td>yes</td>
<td>Java</td>
<td>?</td>
<td>Not able to process the largest ontology</td>
</tr>
<tr>
<td>CEL</td>
<td>yes</td>
<td>LISP</td>
<td>?</td>
<td>for large life science ontologies, implement a polynomial-time</td>
</tr>
</tbody>
</table>

Table 3: Technical table of ontology reasoners
Chapter 6

Conclusions and Outlook

This chapter serves two purposes. Firstly, some insights into realisation of the Semantic Web are given, and secondly, a discussion and trend on outlook is presented.

We have seen that a lot of research have been invested into developing standards and mechanisms for turning Semantic Web into a reality during the last years. There has been progresses in the last few years, but it still is a long way to go. There exist some barriers needed to be addressed. First, the standards of the Semantic Web have a high complexity; thus making the use of tools for creating and maintaining semantic content is necessary. Moreover, the combination of these tools in applications can verify the correctness of ontologies and debug inconsistent ontologies more effectively. Yet this is not enough. There are still insufficient tools on the market for large ontologies. These DLs-based tools can only detect that there is an inconsistency in the ontology. They cannot tell where and how this is caused, so making debugging large ontologies very hard. Second, it is possible that different development communities model the same facts using different ontologies. Therefore the standards of ontologies need to be established for all relevant fields of applications. It will take a long time. At the moment we encourage the research community re-use existing resources before creating new ontologies. Third, the Semantic Web is a top-down approach to establish a technology. These standards are given by the W3C but have not “grown in the web”, and this may waste time in practical application. Hence we should create “seed”
ontologies by bootstrapping approaches in practice.

Based on the above presented materials, there are a number of directions of future research that may be beneficial to the Semantic Web researchers. Such as, SWRL [35] ontology has been accepted by the W3C. It is layered on top of OWL and improves the expressivity of the Semantic Web languages. Hence, it is necessary to keep more tools updated with the technology trend.

Currently, all ontology languages in the Semantic Web stack, such as RDF Schema, OWL and SWRL, are usually interpreted as crisp languages, in which all statements are interpreted to be either true or not. In the real world, the answers of some questions are often not just “yes” or “no”, but express some vague or uncertain knowledge. Hence, fuzzy description logics that integrates fuzzy logic into DLs are proposed [54]. However, there are only a few reasoner accepting fuzzy description logics, e.g., fuzzyDL\(^1\) and FiRE\(^2\), more unfortunately until now there are no fuzzy description logics with hedges. Therefore, the future research direction that is worth pursuing is to incorporate vague or uncertain knowledge with OWL into the Semantic Web and relevant tools.

---

\(^1\)http://gaia.isti.cnr.it/~straccia/

\(^2\)http://www.image.ece.ntua.gr/~nsimou/
Bibliography


[2] National Taiwan University.

http://www.javalobby.org/articles/ajax/.

[4] Semantic Web Road map:


[8] research project: Acquisition for Networks of Negotiating Agents
http://www.iam.ecs.soton.ac.uk/projects/44.html.

[9] research project: Composing and decomposing ontologies: a logic-based ap-
approach
http://www.csc.liv.ac.uk/~dirk/modularity/.
[10] research project: Multimedia Thesaurus and Intelligent Agent support for content based retrieval
   http://www.iam.ecs.soton.ac.uk/projects/64.html.


[12] research project: Explaining Ontology Consequences

[13] research project: Description Logics with Existential Quantifiers and Polynomial Subsumption Problem and Their Applications in Bio-Medical Ontologies

[14] Current projects for University of Stanford

[15] DAML Research Project

[16] Cognitive Agent that Learns and Observes (CALO) Project


[18] Reference description of the DAML + OIL ontology markup language, 2001..


[33] Volker Haarslev, Ralf Möller, and Michael Wessel. Querying the semantic web with racer + nrql.


[38] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.


[50] Biplab K. Sarker, Peter Wallace, and Will Gill. Some observations on mind map and ontology building tools for knowledge management.

