ONTOLOGIES

A Handbook of Principles, Concepts and Applications in Information Systems

by
RAJ SHARMAN
RAJIV KISHORE
RAM RAMESH

Springer's Integrated Series in Information Systems

Chapter 4

USING ONTOLOGIES IN THE SEMANTIC WEB: A SURVEY

Li Ding, Pranam Kolari, Zhongli Ding and Sasikanth Avancha
University of Maryland Baltimore County

Abstract: The Semantic Web is well recognized as an effective infrastructure to enhance visibility of knowledge on the Web. The core of the Semantic Web is "ontology", which is used to explicitly represent our conceptualizations. Ontology engineering in the Semantic Web is primarily supported by languages such as RDF, RDFS and OWL. This chapter discusses the requirements of ontology in the context of the Web, compares the above three languages with existing knowledge representation formalisms, and surveys tools for managing and applying ontology. Advantages of using ontology in both knowledge-base-style and database-style applications are demonstrated using three real world applications.

Key words: Ontology; Semantic Web; survey; tools

1. INTRODUCTION

In philosophy, ontology studies the nature of being and existence. The term 'ontology' is derived from the Greek words "onto", which means *being*, and "logia", which means *written or spoken discourse*. Smith [1] reviewed the studies on the metaphysical aspect of ontology since Aristotle's time, and summarized the essence of ontology as follows: "provide a definitive and exhaustive classification of entities in all spheres of being". In contrast to these studies, Quine's *ontological commitment* [1] [2] drove ontology research towards formal theories in the conceptual world. Computer scientists further extended Quine's work into a new interpretation of ontology as "a specification of a conceptualization" [3].

[1] That is, one is committed as an existing thing when it is referenced or implied in some statements, and the statements are commitments to the thing.

In computer science and information science, knowledge reuse is facilitated by the use of explicit ontology, as opposed to implicit ontology, i.e., knowledge encoded into software systems [4]. Hence, appropriate ontology languages are needed to realize explicit ontology with respect to three important aspects:

- **Conceptualization.** The language should choose an appropriate reference model, such as *entity-relationship model* and *object-oriented model*, and provide corresponding ontology constructs to represent factual knowledge, such as defining the entities and relations in a domain, and asserting relations among entities.
- **Vocabulary.** Besides the semantics, the language should also cover the syntax such as symbol assignment (i.e., assigning symbols to concepts) and grammars (i.e., serializing the conceptualism into explicit representation).
- **Axiomatization.** In order to capture the semantics for inference, rules and constraints are needed in addition to factual knowledge. For example, we can use rules to generate new facts from existing knowledge, and to validate the consistency of knowledge.

On the other hand, web based knowledge sharing activities demand that human and/or machine agents agree on common and explicit ontology so as to exchange knowledge and fulfill collaboration goals. In order to share knowledge across different communities or domains, three requirements should be considered when developing explicit ontology:

- **Extensibility.** In the context of the Web, ontology engineers should be able to develop ontology in an incremental manner: reusing as many existing popular concepts as possible before creating a new concept from scratch. For example, the concept "woman" can be defined as a *sub-class* of an existing concept "person" in WordNet [2] vocabulary. This requirement demands an expressive common reference model as well as distributed symbol resolution mechanisms.
- **Visibility.** Merely publishing knowledge on the Web does not guarantee that it can be readily understood by machines or human users. In order to make knowledge visible on the Web, additional common ontological ground on syntax and semantics is required between information publishers and consumers. This requirement is especially critical to machines since they are not capable of understanding knowledge written in an unfamiliar language.
- **Inferenceability.** Ontology not only serves the purpose of representation, i.e. enumerating factual domain knowledge, but also serves the purpose of computation, i.e., enabling logical inference on facts through axiomatization. Hence, ontology on the Web should provide constructs

[2] See http://wordnet.princeton.edu/.

for effective binding with logical inference primitives and options to support a variety of expressiveness and computational complexity requirements.

The Semantic Web inherits the power of representation from existing conceptualisms, such as *Semantic Networks* [5], and enhances interoperability at both syntactic and semantic levels. It can function as a distributed database or a collaborative knowledge base according to application requirements. In particular, *extensibility* is offered not only by the underlying URI based vocabulary but also by the simple graph data model of *Resource Description Framework* (RDF) [6]. *Visibility* is offered by web based publishing mechanisms (i.e. "Anyone Can Make Statements About Any Resource") which uses URI based vocabulary, XML syntax, RDF graph data model and some common ontology languages. *Inferenceability* is offered by the ontology constructs from *RDF Schema* (RDFS) [7] ontology language and *Web Ontology Language* (OWL) [8] which connect knowledge statement to logical inference at different levels of expressiveness and computational complexity. Formally defined semantics of RDFS and OWL plays an essential role in inferenceability.

This chapter surveys the current deployment status of Semantic Web ontology and corresponding tools, and draws a practical road map on using ontology in the Semantic Web. Section two reviews the evolution of Semantic Web ontology languages by comparing them with existing approaches in database and knowledge representation literature. Section three surveys tools for creating, publishing, extending and reasoning with Semantic Web ontology. Section four surveys storage and integration tools used for applying Semantic Web ontology. Section five discusses three applications to demonstrate the use of ontology in building knowledge-base-oriented and database-oriented applications in the Semantic Web with respect to the three requirements.

2. ONTOLOGY IN THE SEMANTIC WEB

Ontology play an important role in fulfilling semantic interoperability as described in the seminal article on the Semantic Web [9]. W3C has standardized a layered stack of ontology languages that possess the advantages of both knowledge representation (KR) formalisms and conceptual modeling methods for databases. Such standardization activities encouraged creating new ontology and translating pre-existing ontology into the Semantic Web.

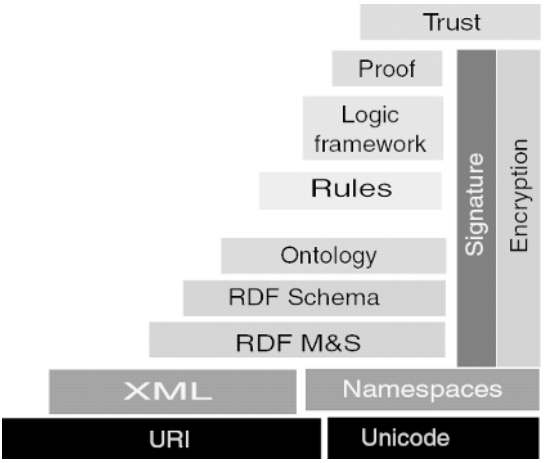


Figure 4-1. The layer cake: enabling standards and technologies for the Semantic Web. Adapted from Tim Berners-Lee (http://www.w3.org/2002/Talks/04-sweb/slide12-0.html)

2.1 Evolution of Semantic Web Ontology Languages

In the Semantic Web layer cake (see Figure 4-1), the semantic part is enabled by a stack of evolving languages: Resource Description Framework (RDF) [6] offers a simple graph reference model; RDF Schema (RDFS) [7] offers a simple vocabulary and axioms for object-oriented modeling; and Web Ontology Language (OWL) [8] offers additional knowledge base oriented ontology constructs and axioms.

Figure 4-2 shows similar evolutionary trends among three paradigms: KR formalisms, conceptual modeling methods for databases, and the Semantic Web. The built-in semantics increases in each paradigm along the vertical axis driven by the demand of porting implicit semantics into explicit representation. For example, *Semantic Networks*, developed between the mid-60s and early 70s, are highlighted by their simple but powerful relational reference model in supporting conceptualization; *Frame Systems* [10], which emerged in the mid-70s, incorporate additional constructs that model classes and instances in a user-friendly manner; *Description Logics* [11], which came out in the 80s as descendents of Semantic Networks and Frame Systems, are highlighted by their formal semantics and decidable inference. Similar evolutions can be observed in the development of the databases and the Semantic Web. RDF was proposed in 1998 as a simple graph model, followed a year later by RDFS. Independent contemporary efforts in DARPA Agent Markup Language (DAML) [3] and Ontology

[3] See http://www.daml.org/.

Inference Layer (OIL) [12] merged into DAML+OIL [13] in 2001 and finally evolved into OWL, which was drafted in 2002 and became a W3C recommendation in 2004.

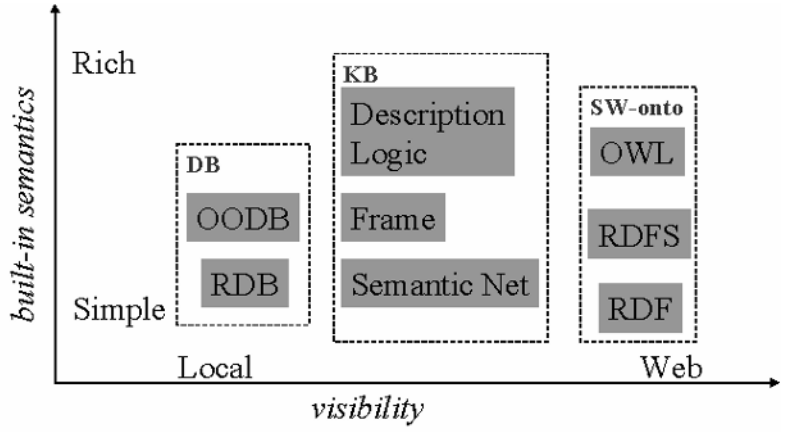


Figure 4-2. A comparison of knowledge representation formalisms (KB), conceptual modeling methods in databases (DB), and Semantic Web ontology languages (SW-onto)

The rapid evolution of Semantic Web ontology languages was enabled by learning from the experiences in developing existing knowledge representation formalisms and database conceptual models, and by inheriting and extending some of their useful features. In particular, the Semantic Web significantly improves visibility and extensibility aspects of knowledge sharing in comparison with the previous approaches. Its URI-based vocabulary and XML-based grammar are key enablers to web scale knowledge management and sharing.

2.2 A Comparison of Ontology Constructs

In order to gain insight into built-in semantics, Table 4-1 summarizes ontology constructs in RDF/RDFS and OWL and compares them with other formalisms in knowledge base (KB) as well as formal models in databases (DB).[4]

[4] Some auxiliary functional constructs are not included in this table: datatype constructs (e.g. rdf:Literal, rdf:XMLLiteral); RDF reification (i.e. rdf:Statement, rdf:subject, rdf:predicate, rdf:object); collections and container (e.g. rdf:List, rdf:Alt, rdf:Bag, and rdf:Set).

Table 4-1. A comparison of ontology constructs*

Cat-1	Cat-2	Ontology Constructs	RDF	RDFS	OWL Lite/DL/Full	RDB	OODB	KB (Frame)	KB (DL)
class	definition	Class		X	X+		X	X	X
		Enumerated Class			–/X/X				O
		Restriction			( )/X/X			O	X
		intersectionOf			( )/X/X			O	X
		unionOf, complementOf			–/X/X			O	X
	axiom	subclassOf		X	X=		X	X	X
		Equality			( )/X/X			O	O
		disjointWith			–/X/X			O	X
relation	definition	Property	X	X=	X+	X	X	O	X
		domain, range		X	X=			O	O
		subPropertyOf		X	X=				
	axiom	(Inverse) Functional			( )/( )/X	X			
		Equality, inverseOf			X				
		transitive, symmetric			()/( )/X				
instance	definition	Type	X	X=	X=		X	X	X
	axiom	(In)Equality			( )		O	O	O

* Cat-1 divides constructs into those describing either a class, a property (i.e., relation), or an individual (i.e., class instance); and Cat-2 divides constructs into those describing concepts/relations or specifying axioms. Table cells are marked to show how an ontology construct is supported by an ontology language or formalism: 'X' means fully supported; 'X=' means fully support via inheritance, 'X+' means extended fully support, '( )' means supported with restriction; '-' means not supported; and 'O' means optionally supported.

2.2.1 RDF

RDF offers a simple graph model which consists of nodes (i.e. resources or literals) and binary relations (i.e. statements). It is a type of Semantic Network and is very similar to the *Relational Model* [14]. Such a simple model embodies a small amount of built-in semantics and offers great freedom in creating customized extensions; however, an extended or specialized semantic network is usually required in practice. John Sowa identifies six categories of semantic networks based on relation semantics [15]: (i) *Definitional networks*, which build taxonomies for conceptualisms with inheritance (subclass) and membership (instance) relations; (ii) *Assertional networks*, which represent cognitive assertions about the world

with modal operators; (iii) *Implicational networks*, which focus on implication relations, e.g. belief network; (iv) *Executable networks*, which focus on temporal dependence relations, e.g. flowchart, PetriNet; (v) *Learning networks*, which focus on causal relations encoded in numerical value, e.g. neural network; (vi) *Hybrid networks*, which combine features of previous types. In the Semantic Web, most ontology are defined using RDF(S)/OWL and thus fall in the first category; the second category (assertional networks) emerges in the context of sharing instance data and evaluating trustworthiness of such data, e.g., [16, 17, 18, 19, 20, 21]; and the third category (implicational networks) gains interests in ontology mapping study [22, 23]. A variation of definitional networks is natural language encyclopedia such as dictionaries and thesaurus which uses a different set of relations rather than class-property relation. WordNet and Simple Knowledge Organization System (SKOS, http://www.w3.org/2004/02/skos/) are their representative Semantic Web versions respectively.

2.2.2 RDFS

Under the influence of *Frame Systems* and the *Object Oriented Model*, RDFS has been used to augment RDF to provide better support for definition and classification [24]. These models organize knowledge in a concept-centric way with descriptive ontology constructs (such as frame, slot, and facet) and built-in inheritance axioms. Frame Systems enable users to represent the world at different levels of abstraction with the emphasis on entities, and this aspect makes it quite different from the planar graph model offered by most semantic networks. In addition to inheriting basic features from Frame Systems, RDFS provides ontology constructs that make relations less dependent on concepts: users can define relations as an instance of *rdf:Property*, describe inheritance relations between relations using *rdfs:subPropertyOf*, and then associate defined relations with classes using *rdfs:domain* or *rdfs:range*.

2.2.3 DAML+OIL and OWL

DAML+OIL and OWL extend RDFS and emphasize support for richer logical inference. Besides inheriting advantages from Frame Systems, these ontology languages provide a rich set of constructs based on formal Model Theoretic Semantics[5]. Three variants of OWL trade off computational complexity and the expressiveness of ontology constructs.

[5] See http://www.w3.org/TR/rdf-mt/ (RDFS), http://www.w3.org/TR/owl-semantics/ (OWL).

- *OWL-Lite* is the simplest variant for building a basic frame system (or an object oriented database) in terms of class, property, subclass relation, and restrictions. OWL-Lite does not use the entire OWL vocabulary and some OWL terms are used under certain restrictions.
- *OWL-DL* is grounded on Description Logics, and focuses on common formal semantics and inference decidability. Description logics offer additional ontology constructs (such as conjunction, disjunction, and negation) besides class and relation, and have two important inference mechanisms: subsumption and consistency. Horrocks and Sattler [25] argued that basic inference in most variations of Description Logics is decidable with complexity between polynomial and exponential time. The strong Set Theory background makes Description Logics suitable for capturing knowledge about a domain in which instances can be grouped into classes and relationships among classes are binary. OWL-DL uses all OWL ontology constructs with some restrictions.
- *OWL-Full* is the most expressive version of OWL but it does not guarantee decidability. The biggest difference between OWL-DL and OWL-Full is that class space and instance space are disjointed in OWL-DL but not in OWL-Full. That is, a class can be interpreted simultaneously as a set of individuals and as an individual belonging to another class in OWL-Full. The entire OWL vocabulary can be used in without any restrictions in OWL-Full.

2.3 Swoogle's Survey of Semantic Web Ontology

This subsection surveys ontology with emphasis on the Semantic Web context, in contrast to prior surveys on ontology development [26, 27]. According to a recent report by Swoogle[6], a search engine that indexes the Semantic Web on the Web, over ten thousand Semantic Web ontology have been discovered on the Web. Table 4-2 (a, b) lists some well populated Semantic Web ontology discovered by Swoogle. Existing Semantic Web ontology can be classified into the following four major categories (without clear-cut boundaries): meta-ontology, comprehensive upper ontology, systematic domain specific ontology, and simple specialized ontology.

Table 4-2a. Usage of Semantic Web meta-ontology (Swoogle, July 2005)

Ontology prefix	Namespace URI	# of Docs. Populated
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	382*K*
rdfs	http://www.w3.org/2000/01/rdf-schema#	82*K*
owl	http://www.w3.org/2002/07/owl#	64*K*
daml	http://www.w3.org/2001/03/daml+oil#	5*K*

[6] See http://swoogle.umbc.edu.

Table 4-2b. Popular Semantic Web ontology (Swoogle, July 2005)

Ontology prefix	Namespace URI	# of Docs. Populated
dc	http://purl.org/dc/elements/1.1/	250*K*
rss	http://purl.org/rss/1.0/	165*K*
admin	http://webns.net/mvcb/	130*K*
sy	http://purl.org/rss/1.0/modules/syndication/	90*K*
foaf	http://xmlns.com/foaf/0.1/	77*K*
cc	http://web.resource.org/cc/	74*K*
content	http://purl.org/rss/1.0/modules/content/	60*K*
trackback	http://madskills.com/public/xml/rss/module/trackback	56*K*
iw	http://inferenceweb.stanford.edu/2004/05/iw.owl#	47*K*
bio	http://purl.org/vocab/bio/0.1/	35*K*
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	25*K*
vCard	http://www.w3.org/2001/vcard-rdf/3.0#	20*K*

2.3.1 Meta-Ontology

The ontology languages, namely RDF, RDFS, DAML+OIL and OWL, are in fact meta-ontology themselves; and their instances are Semantic Web ontology. Such meta-ontology offers a small vocabulary and corresponding axioms as the building blocks for any conceptualisms, and they are backed by inference engines with built-in support for their ontology constructs and axioms. For example, a RDFS inference engine can understand the semantics of *rdf:subClassOf* and infer RDF triples by propagating *rdf:type* statement through sub-class relations. Such ontology only provides necessary parts for the reference model without considering any domain concepts.

There are also some additional candidate ontology languages. In order to represent the semantics of rules, rule/policy languages have been proposed, such as Semantic Web Rule Language (SWRL) [7], which is a combination of OWL and RuleML, and Rei declarative policy language [28]. In additional to the object-oriented constructs provided in RDF(S) and OWL, ontology constructs for thesaurus like concept organization (e.g. concept, narrower-concept, and related-concept) have been modeled in SKOS.

2.3.2 Comprehensive Upper Ontology

Upper ontology provides a high level model about the world using the meta-ontology. Currently, Semantic Web researchers are working to translate existing upper ontology, such as Cyc [29], WordNet [30, 31], OntoSem [32], and IEEE's Standard Upper Ontology (SUO) [33], into

[7] See http://www.daml.org/2003/11/swrl/.

RDF(S) or OWL versions. OpenCyc (http://www.opencyc.org/) has published a 700MB OWL files encoding part of the CYC ontology. There is also a RDFS version of WordNet ontology using the namespace http://xmlns.com/wordnet/1.6/, and a W3C's task force [8] has been formed recently aiming at better RDF(S)/OWL based representation of WordNet. OntoSem is being translated into OWL [34, 35].

2.3.3 Systematic Domain Specific Ontology

Unlike upper ontology which require agreements across multiple domains, *domain specific ontology* have been developed to build systematic vocabulary for certain domains long before the inception of the Semantic Web, e.g. legal ontology [36], gene ontology [37], chemical ontology [38], bio ontology [39], and spatial ontology [40]. Again, the Semantic Web makes it possible to improve the visibility of such domain ontology; hence, translation efforts such as building an RDF version of CIA world fact book are ongoing. Domain ontology can also contain some well-known class instances besides class/property definition, e.g. airport ontology not only defines the class "airport", but also enumerates all three-letter airport codes.

2.3.4 Simple Specialized Ontology

One difficulty with comprehensive or systematic ontology is that they are usually too big to use. For example, no existing ontology inference engine can store and use the complete OpenCyc ontology which has over 60,000 terms and is stored in a 700MB file. Hence, much simple specialized Semantic Web ontology have been developed to overcome this difficulty by concentrating on a set of basic and commonly-used concepts. Such ontology is often used as interchange languages in knowledge sharing.

Dublin Core (http://dublincore.org/) brought about a series of ontology for document metadata, e.g., the well-known RDFS based ontology – Dublin Core Metadata Element Set ontology. RSS news digest ontology (including rss, sy, trackback, and content as listed in Table 4-2 is driven by the blogging community and has now become one of the most popular domain ontology. W3C is also driving the Friend-Of-A-Friend (FOAF) [9] ontology for person information. The Inference Web ontolog [10] focuses on explicit representation of justification steps produced by inference engines. The Creative Commons ontology aims at recording copyright related information. Similarly,

[8] See http://www.w3.org/2001/sw/BestPractices/WNET/tf.html.
[9] See http://www.foaf-project.org.
[10] See http://inferenceweb.stanford.edu.

ontology such as 'geo', 'vCard', and 'admin' have been developed with small vocabulary sets specialized to capture relevant domain information.

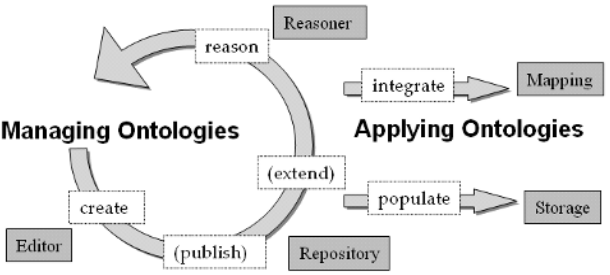


Figure 4-3. Ontology tools for managing ontology and applying ontology. The 'publish' and 'extend' steps are parenthesized to indicate they are optional

3. SEMANTIC WEB ONTOLOGY TOOLS

Our discussion thus far has shown how ontology play a critical role in Semantic Web applications. Effective enabling tools are needed in order to implement Semantic Web ontology. Figure 4-3 depicts typical steps in managing ontology, i.e., create, publish, extend, and reason; and two common scenarios in applying ontology: populating instances of ontology and integrating information encoded by different ontology. Accordingly, five tool classes are placed close to relevant steps or scenarios: tools for managing ontology are covered in this section; and tools for applying ontology are covered in Section four.

3.1 Ontology Editors

A good editor can save a significant amount of time when developing ontology by helping ontology engineers focus on the semantics without worrying much about syntactic organization. This section offers a brief introduction to some popular and Semantic Web related ontology editors for collaborative or independent ontology development. A more comprehensive survey can be found in [41].

Protege [42] provides a standalone ontology development environment. It is highlighted by its syntax grammar independent user interface and pluggable infrastructure. It is suitable for independent ontology development and has a large user community.

SWOOP [43] takes advantage of both Protege and Ontolingua [44] (a web-based environment for editing, publishing, and sharing ontology

developed before the Semantic Web) and provides a convenient web-based ontology browsing, editing, debugging [45] and publishing interface.

3.2 Ontology Repositories

Although the Web improves the visibility of centralized ontology development, it is hard to achieve a universal ontology for everything (e.g. Cyc) due to huge space complexity. Hence, distributed ontology development is preferred in the Semantic Web, i.e., small ontology are authored by different sources in an incremental fashion. To reuse existing ontology, effective web based tools are in great need to browse, search and navigate distributed ontology. Table 4-3 compares some popular repositories for publishing and searching ontology on the Web, and their technical highlights are detailed below.

Table 4-3. A comparison of Semantic Web ontology access services

	DAML ontology Library	SchemaWeb	Ontaria	Semantic Web Search	Swoogle
Indexed documents/ ontology	282/282	203/203	N/A	N/A	>1M / >10K
search ontology	listAll	listAll; Full-text;	keyword	use resource search	keyword
search document	no	no	yes	use resource search	yes
search class/ property	keyword	triple pattern	keyword	keyword	keyword; alphabetical index
search RDF resource	no	triple pattern	N/A	keyword	no
navigation	no	no	yes	no	yes
annotation	by user	by user	no	no	auto digest
auto discovery	no	no	no	yes	yes

- **DAML Ontology Library** (http://www.daml.org/ontology/) indexes user submitted ontology and provides browse/search services. It organizes ontology by their URI, users' annotations supplied during ontology submission (e.g. submission date, keyword, open directory category, funding source, and submitter's organization), the defined class/property, or the used namespace. Users can run sub-string queries over a defined class/property.
- **SchemaWeb** (http://www.schemaweb.info/) provides services similar to DAML ontology library with better human/machine user interface (i.e. both HTML and web service interface). It adds more services: (i) for human user, it provides full-text search service for indexed ontology, and

a customizable resource search interface by letting users specify triple pattern; (ii) for machine agents, it searches the "official" ontology of a given namespace or the resource with user specified triple pattern; it also navigates RDF graph through RDFS properties (i.e. subClassOf, subPropertyOf, domain, range), and publishes RSS feeds about new ontology submissions.

- **W3C's Ontaria** (http://www.w3.org/2004/ontaria/) stores RDF documents (including ontology) and provides search/navigation services in the repository. It allows a user to (i) browse a RDF file as a list of triples, a list of used properties, or a list of populated classes, and (ii) browse relations between RDF files.
- **Semantic Web Search** (http://www.semwebcentral.org/) provides an object oriented view of the Semantic Web, i.e. it indexes instances of well-known classes including *rdfs:Class*, *rdf:Property*, *foaf:Person*, and *rss:Item*. It partially supports ontology search by finding instances of *rdfs:Class* and *rdf:Property*; however, its search results are biased to terms from the namespace of *WordNet 1.6*.
- **Swoogle** (http://swoogle.umbc.edu) indexes millions of Semantic Web documents (including tens of thousand of ontology). It enables users to search ontology by specifying constraints on document metadata such as document URLs, defined classes/properties, used namespaces, and RDF encoding. Moreover, it provides detailed metadata about ontology and classes/properties in an object oriented fashion. It has an ontology dictionary that enables users to browse the vocabulary (i.e. over 150KB URIrefs of defined/used classes and properties) used by Semantic Web documents, and to navigate the Semantic Web by following links among classes/properties, namespace and RDF documents. In addition, it is powered by automatic and incremental Semantic Web document discovery mechanisms and updates statistics about the use of ontology in the Semantic Web on a daily basis.

3.3 Ontology Language Reasoners

An ontology construct conveys descriptive semantics, and its actionable semantics is enforced by reasoners (i.e. inference engines). Table 4-4 introduces several popular reasoners by comparing how they support the inferenceable semantics of Semantic Web ontology languages. A detailed developers' guide is available online[11], and experimental evaluation can be found in W3C's OWL Test Cases report[12].

[11] See http://www.wiwiss.fu-berlin.de/suhl/bizer/toolkits/.
[12] See http://www.w3.org/2003/08/owl-systems/test-results-out.

Table 4-4. Capabilities of reasoners for the Semantic Web ontology*

	OWLJessKB	JTP	Jena	F-OWL	FaCT++	Racer	Pellet	TRIPLE	Sweet Rules
RDFS	()	()	+	()	()	()	()	()	-
OWL-Lite	()	()	()	()	+	()	()	()	-
OWL-DL	-	()	()	()	()	()	()	()	-
OWL-Full	-	()	()	()	-	-	-	()	-
RuleML	-	-	-	-	-	-	-	-	+
Language	Java	Java	Java	Java	C++	Lisp	Java	Java	Java

* Here, '+' means full-support, '-' means no support, and '()' means partial support.

- **OWLJessKB** (http://edge.cs.drexel.edu/assemblies/software/owljesskb/) is the descendent of DAMLJessKB [46] and is based on the Jess Rete inference engine.
- **Java Theorem Prover (JTP)** (http://www.ksl.stanford.edu/software/JTP/), developed at Stanford University [47], supports both forward and backward chaining inference using RDF/RDFS and OWL semantics[13].
- **Jena** (http://jena.sourceforge.net/), developed at HP Labs at Bristol [48], is a popular open-source project. It provides sound and almost complete (except for blank node types) inference support for RDFS. Current version of Jena also partially supports OWL inference and allows users to create customized rule engines.
- **F-OWL** (http://fowl.sourceforge.net/), developed at UMBC [49], is an inference engine which is based on Flora-2[14].
- **FaCT++** (http://owl.man.ac.uk/factplusplus/), developed at the University of Manchester [50], is the descendent of FaCT [51] reasoning system. It provides full support for OWL-Lite. Its future releases aim at providing complete support for OWL-DL reasoning.
- **Racer** (http://www.sts.tu-harburg.de/r.f.moeller/racer/) is a description logic based reasoner [52]. It supports inference over RDFS/DAML/OWL ontology through rules explicitly specified by the user.
- **Pellet** (http://www.mindswap.org/2003/pellet/), developed at the University of Maryland, is a 'hybrid' DL reasoner that can deal both TBox reasoning as well as non-empty ABox reasoning [53]. It is used as the underlying OWL reasoner for SWOOP ontology editor [43] and provides in-depth ontology consistency analysis.

[13] See http://www.ksl.stanford.edu/software/JTP/doc/owl-reasoning.html.
[14] Flora-2 is an object oriented language with Frame System background.

- **TRIPLE** (http://triple.semanticweb.org/), developed by Sintek and Decker [54], is a Horn Logic based reasoning engine (and a language) and uses many features from F-logic. Unlike F-logic, it does not have fixed semantics for classes and objects. This reasoner can be used by translating the Description Logics based OWL into a language (named TRIPLE) handled by the reasoner. Extensions of Description Logics that cannot be handled by Horn logic can be supported by incorporating other reasoners, such as FaCT, to create a hybrid reasoning system.
- **SweetRules** (http://sweetrules.projects.semwebcentral.org/) is a rule toolkit for RuleML. RuleML is a highly expressive language based on courteous logic programs, and provides additional built-in semantics to OWL, including prioritized conflict handling and procedural attachments. The SweetRules engine also provides semantics preserving translation between various other rule languages and ontology (implicit axioms).

4. APPLYING SEMANTIC WEB ONTOLOGY

In many practical scenarios, ontology tools introduced in the previous section are not sufficient for application development on the Semantic Web. The rest of this section surveys two important issues: (i) providing inference support in populating and storing instances of Semantic Web ontology in large scale applications; (ii) mapping concepts from different ontology in ontology based information integration.

4.1 Storing Ontology Instances

Though technology in large scale relational models is relatively mature, the inferenceability feature of ontology introduces additional requirements on storing instances of ontology. Users should now be able to access the asserted knowledge as well as the inferred knowledge which can be derived by ontology based inference. In the Semantic Web, instances of ontology (i.e, knowledge represented in RDF triples) are stored in so-called *triple stores* or *RDF databases*.

There are three alternative strategies to manage inferred knowledge in triple store as influenced by logic inference [55].

- **Forward Chaining** applies entailment rules as soon as RDF triples have been added into a triple store. This approach eliminates run-time inference by enumerating and storing all possible knowledge; hence it will result in fast query response at the cost of increased load time and storage space. Unfortunately, this approach is not so promising since (i) there is no guarantee that the inferred triples will be queried in the future

and (ii) the additional storage space for inferred triples can be prohibitively large and impose overhead in access.

- **Backward Chaining** applies entailments rules when the triple store is queried. This approach performs run-time inference without the need of storing inferred knowledge; hence it will result in short load time at the cost of increased query response time. Since query response time is an important benchmark of ease-of-use, too slow response time will decrease users' adoption.
- **Hybrid Inference** combines both forward and backward chaining so as to avoid the disadvantages of both.

There are two well-known storage options, namely in-memory storage and persistent storage. Their performance has been evaluated by comparing the efficiency of load/save operations [56], soundness and completeness of inference [57], and scalability [58]. Table 4-5 compares basic features of some popular triple stores with the emphasis on the persistent storage option since knowledge in the Semantic Web is expected to be in large amount. In the following text, the term 'model' is used to signify the RDF graph including both asserted and inferred triples.

Table 4-5. Capabilities of persistence triple stores *

	Query Language	Level of Inference	Inference Strategy
Jena	RDQL	RDFS, OWL(partial)	F, B, H
RSSDB	RQL	RDFS	-
Kowari	iTQL, RDQL	Explicit Rules	F
Sesame	SeRQL	RDFS	-
3Store	OKBC, RDQL	RDFS	H
Instance Store	Racer, FaCT++ based	OWL-Lite, OWL-DL	-
DLDB	conjunctive KIF	OWL-DL	F

* The three kinds of inference strategies are abbreviated as follows: 'F' for Forward Chaining, 'B' for Backward Chaining and 'H' for Hybrid. '-' is used when corresponding information is not available.

- **Jena** [59] offers both in-memory and persistent storage options. It provides physical data independence through its Java based data access library which hides the physical storage details; hence there is not much difference between accessing persistence store or in-memory store.
- **RSSDB** [60] implements persistent storage option. In addition to being a general triple store, it improves data storage efficiency by storing instances of a class in a specialized (Object-Relational) table at the expense that it assumes that domain ontology are fixed and have defined classes with significant amount of instances.

- **Kowari** (http://www.kowari.org/) implements persistent storage using flat files instead of conventional database products. It allows users to explicitly associate various inference rules (e.g. axioms) with the asserted triples, and separates the storage of asserted triples from that of the inferred triples.
- **Sesame** (http://www.openrdf.org/) [61] implements persistent storage option using Forward Chaining and RDFS level inference, i.e., it enumerates and stores all inferred triples according to RDFS semantics and domain ontology.
- **3Store** [55] supports hybrid inference mechanisms. It classifies inference axioms into those which generate comparatively fewer entailments and apply forward chaining inference on them, and uses backward chaining to handle the rest.
- **Instance store** (http://instancestore.man.ac.uk/) [62] provides Description Logics level persistent storage, and it relies on the FaCT++/Racer inference engine.
- **DLDB** [63] supports persistent triple store by explicitly using the FaCT reasoner which support Description Logics inference. Similar to Sesame, entailment is pre-computed based on the assumption that ontology change less frequently. Similar to RSSDB, it uses a class-centric view for efficient query processing.

The query languages provided and the ontology languages supported by the above triple stores are listed in Table 4-5. A detailed discussion can be found in [64]. In addition, W3C is standardizing SPARQL [65], a new common query language for triple stores.

Existing triples stores are weak in the scalability aspect, i.e., in-memory approaches are limited by the size of main memory and persistent stores are not sufficiently scalable. Two research directions are tackling this issue. First, distributed storage approach [66], Peer-to-Peer system based approach [67], and efficient index based approach [68] have been proposed in contrast to the current centralized storage. Second, researchers are also exploring efficient hybrid inference strategies which prevent full exploration of search space and keep the inference overhead in processing users' queries in an acceptable range.

4.2 Ontology-Based Information Integration

The Semantic Web puts the onus of ontology creation on the user by providing common ontology languages such as RDF(S) and OWL. However, ontology defined by different applications or agents usually describe their domains in different terminologies, even when covering the same domain. The semantic-level heterogeneity between two information sources refers to

the use of conflicted or mismatched terms about concepts in their corresponding ontology, which can be identified into one of the following categories: (i) *ambiguous reference* – the same term (i.e., the symbolic identifier of a concept in an ontology) means differently in different ontology; (ii) *synonymous reference* – two terms from different ontology have the same meaning; (iii) *one-to-many matching* – one term from one of the ontology matches [15] to several terms of the other ontology; (iv) *uncertain matching* – one term from one of the ontology has similar but not exactly the same meaning to any terms of the other ontology; and (v) *structural difference* – two terms with the same or similar meaning are structured differently in different ontology (e.g., different paths from their respective root concepts).

In order to support ontology-based information integration, tools and mechanisms are needed to resolve the semantic heterogeneity problem and align the terms in different ontology. This section reviews the literature about the existing works in this topic, which is grouped into five different research directions in tackling the problem.

- **A Centralized Global Ontology.** Enforcing a centralized global ontology prevents semantic heterogeneity since no more ontology exists and everyone is using the same ontology. However, this approach is obviously impractical since (i) the creation and maintenance of such ontology is usually prohibitively expensive and (ii) it is usually impractical to develop ontology with consent from the user community at large.
- **Merging Ontology.** Merging different ontology into a unified one is another natural approach to semantic integration when those ontology overlap significantly over a common domain. There are many heuristics to merge two terms, such as (i) linguistic heuristics which uses term spelling or additional natural language processing (NLP) techniques with manual validation, e.g., *FCA-MERGE* [69, 70], (ii) syntactic and semantic heuristics, e.g., *PROMPT* [71] [16] and *Chimaera* [72], and (iii) hybrid approaches [73]. However, this approach is usually costly and not scalable. The merging procedure has to restart from scratch when any of the input ontology has been modified. When merging a large number of ontology, the merging result may not always meet application.
- **Mapping Ontology.** Building a set of mappings (or matches) between two ontologies is an alternative way to merging ontology. A mapping between two terms from two different ontologies conveys the fact that the terms have similar or same meaning. Besides manually specifying

[15] A subject 'matches' an object if they refer to exactly the same concept.

[16] It initializes term-matching suggestions using linguistic similarity among class names, and then updates suggestions by resolving newly detected syntactic and semantic conflicts.

mappings, there are some semi-automated methods such as: (i) lexical similarity analysis on linguistic or lexical ontology [74, 75] such as WordNet, Cyc, and SENSUS; (ii) textual description analysis, which assigns a set of relevant documents to each term so as to capture the meaning of the term, measures similarity between terms using machine learning based text classification techniques, and searches for mappings based on the similarity matrix obtained, e.g., *CAIMAN* [76], *OntoMapper* [77], and *GLUE* [78, 79, 80]; (iii) ontology algebra and articulation, e.g., *ONION* [81], which is semi-automatic, with good scalability, easy to maintenance, but slow; (iv) information flow and channel theory based approach [82]; (v) structural analysis, i.e., 'similarity flooding' – a graph matching algorithm based on fixpoint computation [83]; and (iv) hybrid heuristics, sometimes combined with the structural information of the ontology taxonomy, e.g., *Anchor-PROMPT* [84] and *PROMPTDIFF* [85]. A brief survey of existing approaches is provided by [86], however, most of these approaches only study exact mappings, without taking the degree of uncertainty [17] into consideration [18]. Since semantic similarities between concepts can be easily represented probabilistically (but not logically), Bayesian Networks (BNs) [87] stand out as a natural choice in tackling this problem: (i) Mitra et al. [88] improve existing mapping results by applying a set of meta-rules to capture the structural influence and the semantics of ontology relations; (ii) Ding et al. [89] and Pan et al. [90] proposed a principled methodology by first translating the source and target ontology into BNs, and then mapping the concepts from the two ontology based on evidential reasoning between the two translated BNs.

- **Ontology Translation.** Given two ontology, ontology translation is to translate one of the ontology into a target ontology which uses the representation and semantics of the other ontology, sometimes with the help of an intermediate shared ontology. Based on a set of defined rules and transformation operators, *Ontomorph* [91] offers syntactic rewriting and semantic rewriting to support the translation between two different knowledge representation languages. *OntoMerge* [92], an online ontology translation system [19] based on ontology merging (which requires a set of ontology bridging axioms produced manually by domain experts) and automated reasoning, achieves term translations using a first order

[17] It is often the case that a concept defined in one ontology can only find partial matches to one or more concepts in another ontology.

[18] Note that the methods in (ii) fail to completely address uncertainty in mapping since the degree of similarity found between concepts will not be considered in further reasoning.

[19] See http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html.

theorem prover built on top of PDDAML (PDDL-DAML Translator) [20] (based on Jena) and OntoEngine [21] (an inference engine based on JTP), in either forward or backward chaining way. Ontology translation takes a further step after mapping or merging, and is one of the most difficult tasks towards information integration.

- **Runtime Ontology Resolution.** Semantic differences can arise in run-time interaction in a multi-agent environment since it is impractical to restrict all agents to use the same ontology. Merging, mapping, or translating ontology are impractical too since they are usually offline approaches which need to be done before the deployment of the multi-agent system. One family of approaches [93, 94, 95, 96] is inspired by language games, where agents identify and resolve ontology conflicts through incremental interpretation, clarification, and explanation by negotiating with one another when semantic differences have been detected. An alternative approach utilizes approximate classification methods for semantic-preserving context transformations, such as rough set theory, fuzzy set, or probabilistic Bayes' Theorem [97, 98].

Since the interoperability between different knowledge systems or agents relies on their full understanding of the terminologies used by the peers, the resolution of semantic heterogeneity between different information sources is necessary and important. Hence, this aspect currently attracts significant attention from the Semantic Web research community.

5. USING SEMANTIC WEB ONTOLOGY

The semantics conveyed by ontology can be as simple as a database schema or as complex as the background knowledge in a knowledge base. By using ontology in the Semantic Web, users can leverage the advantages of the following two features: (i) data is published using common vocabulary and grammar; and (ii) the semantic description of data is preserved in ontology and ready for inference. This section presents three real-world Semantic Web based applications to show the different roles of Semantic Web ontology played in different context.

The first application is called *semantic service discovery*, which builds an extensible ontology to describe the various data services in ad-hoc networks, and uses ontology to reason the capability of sensors. It is highlighted by the *extensibility* aspect of the Service ontology. The second application is called *ontology based personal profile integration*, which builds a web scale database for personal profiles. It is highlighted by the *visibility* aspect of

[20] See http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator1.html.
[21] See http://projects.semwebcentral.org/projects/ontoengine/.

FOAF ontology. The third application is called *description logic reasoning for adaptive sensors*, which infers sensor states using the axioms in OWL-DL. It is highlighted by the *inferenceability* aspect of Sensor State ontology.

5.1 Semantic Service Discovery

Avancha et al. [99] have used ontology to represent the profile and infer the capability of services in ad-hoc networking environment like Bluetooth.

5.1.1 Service Ontology

Ontology-based service description is superior to UUID-based descriptions [100] because of the following merits of the former: (i) it enables extensible and richer description about the services, entities and the relationships between entities in application domain; (ii) it supports inferring implied knowledge from the asserted descriptions; and (iii) it captures domain knowledge in the explicitly represented ontology (instead of in source code) and thus make domain ontology independent to source code.

The Service Ontology is created through the following two steps.

1. *Choosing an appropriate Semantic Web ontology language.* In order to achieve successful semantic service discovery, a simple but powerful model is needed for describing domain knowledge. It should be lightweight - easy to parse, easy to manipulate and easy for a reasoning engine to use. It should be extensible, so that any organization or person can create classes or properties that can be added to ontology. It should be scalable enough to handle huge number of resources in a given ontology. Hence, RDFS was chosen as the ontology language because it meets all the above requirements and no advanced feature from OWL is needed in this application.
2. *Building the service ontology which captures background knowledge.* The root of the service ontology is a class called *Service*. It has one subclass called *AdHocNetworkService*. Specific services are described as subclasses of the latter. The Service class has two properties – *ServiceCost* and *ProvidedBy*. The latter "points" to a class called *ServiceProvider* that contains *ProviderName* and *ContactURI* as its properties. Every property of the *AdHocNetworkService* class, except *MemoryCapacity*, is associated with a *value* and a *priority*. The priority field is used to decide the ordering of properties, so that the highest priority property (specified by the client in the query or assigned by the server) is used to determine the correct service instance to match against. The client is allowed to leave the *priority* field(s) unspecified. The client may also leave the *value* field(s) unspecified. In such cases, the server uses a set of predefined priority values for the properties.

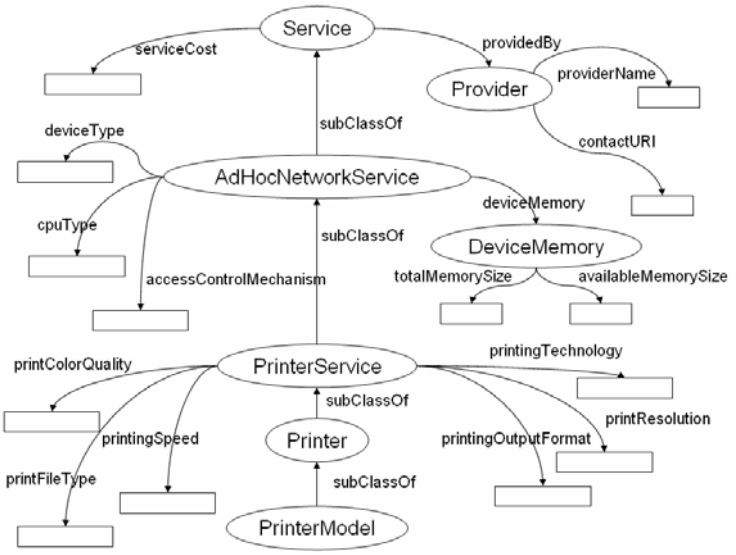


Figure 4-4. The service ontology. Oval nodes represent classes (concepts) and rectangular nodes represent literal values. Labels on the arcs represent properties between concepts

Figure 4-4 shows a graphical representation of the Service Ontology using an example *PrinterService*. The *priority* and *value* property-types for all of the property-types of the *AdHocNetworkService*, *PrinterService* and *Printer* subclasses are not shown for conciseness. The *PrinterService* subclass has six property-types and the *Printer* subclass has only one. However, it is emphasized that the *Printer* subclass inherits all property-types from its ancestors. Therefore, all the property-types can be associated with values at the level of the *Printer* subclass.

5.1.2 Applying Service Ontology and Evaluation

By reusing the service ontology written in RDFS, *extensibility* feature can be best demonstrated when incorporating new devices or services, resources, capabilities and constraints into the ac-hoc network. Description of new service can be greatly reduced by inheriting property-types using sub-class relation. In addition, extending domain knowledge does not require any code level changes.

By using inference engine which supports RDFS, *inferenceability* feature can be demonstrated when matching service request with service description. The inference engine combines the asserted and implied domain knowledge into the complete service descriptions, and thus gets better matching results.

Readers interested in the details of the approach and experimental results are referred to [101].

5.2 Ontology-based Personal Profile Integration

Recently, the "Friend of A Friend" (FOAF) project has gained attention from both academic and industrial communities. Aiming at publishing information about people using machine understandable language, the FOAF project is highlighted by the following features: (i) publishing personal profile with better visibility; (ii) enforcing unique reference to a person on the Web and supporting integration of partial data from difference sources; and (iii) representing and facilitating large scale social networks on the Web.

5.2.1 FOAF Ontology

The core of FOAF project is the FOAF ontology which is identified by namespace URI *http://xmlns.com/foaf/0.1/*. The core concepts of FOAF ontology are shown in Figure 4-5 an agent could be a person, a group, or an organization, among which person is the most used; a *foaf:Person* usually has her name, mbox, depiction(personal photo), homepage, and etc.; and social network could be captured by the *foaf:knowns* property between two instances of *foaf:Person*.

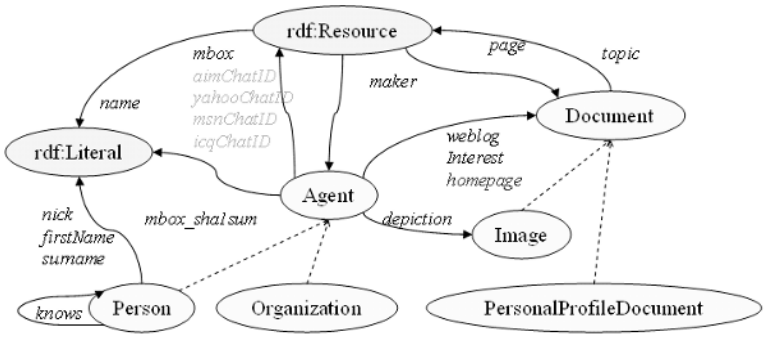


Figure 4-5. Important classes and properties in FOAF Ontology. Solid arcs refer to properties linking domain-class to range-class, and dashed arcs refer to subClassOf relation

5.2.2 Integrating FOAF Personal Profile

The simple and commonly adopted FOAF ontology make it possible to build a distributed database for personal information. Among millions of Semantic Web documents discovered by Swoogle [102], a large portion of them contribute a significant amount of instances of *foaf:Person*. An interesting observation shows that the person "Tim Finin" has been mentioned by some Semantic Web documents from UMBC and CMU, and

several of his email addresses have been mentioned in different documents. According to the entity-equality semantics of FOAF ontology, the RDF graphs in those RDF documents can be merged into a more complete personal profile. The integration procedure is described as the following steps:

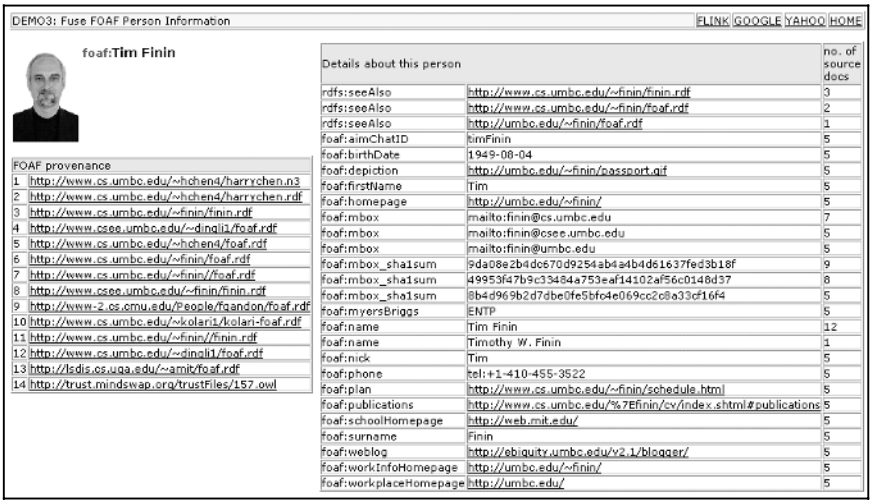
1. Swoogle discovers and indexes Semantic Web documents on the Web.
2. Search Swoogle for all Semantic Web documents that populate instances of *foaf:Person*.
3. Parse those documents and store all instances of *foaf:Person* in a triple store.
4. Merge instances using the following heuristics: (i) two instances with the same URIref; (ii) compare the identities of instance obtained from (inverse) functional properties; and (iii) use *owl:sameAs* assertions over instances. In practice, most instances are anonymous (i.e., without URI identifier) and the second heuristic is heavily used. The merge could be implemented by computing connected component on a graph where each node is a class instance and each arc conveys one instance-equivalence relation.
5. Output each connected component as a merged personal profile.

Figure 4-6 shows the merged profile for "Tim Finin". It should also be noted that such merging has to consider possible errors in the source documents. The common error is caused due to the wrong usage of inverse functional properties. For example, a Semantic Web document [22] mistakenly assign the same mbox_sha1sum to thousands of instances of *foaf:Person*. More observation could be found in [103].

5.2.3 Evaluation

The *visibility* of FOAF ontology based knowledge is boosted significantly through URI-based vocabulary and RDF/XML based serialization. On the publisher side, many social network websites, such LinkedIn.com, Okurt.com, and LiveJournal.com, have adopted FOAF ontology in publicizing their users' profile and millions of Semantic Web documents are published on such websites [103]. On the information consumer side, many XSL based tools, such as *Foaf Explorer* and *Foafnaut* (http://www.foafnaut.org), have been developed to visualize FOAF data in a user friendly format.

[22] See http://blog.livedoor.jp/rusa95/foaf00756.rdf.



DEMO3: Fuse FOAF Person Information — FLINK | GOOGLE | YAHOO | HOME

foaf:**Tim Finin**

FOAF provenance	
1	http://www.cs.umbc.edu/~hchen4/harryrchen.n3
2	http://www.cs.umbc.edu/~hchen4/harryrchen.rdf
3	http://www.cs.umbc.edu/~finin/finin.rdf
4	http://www.csee.umbc.edu/~dingli1/foaf.rdf
5	http://www.cs.umbc.edu/~hchen4/foaf.rdf
6	http://www.cs.umbc.edu/~finin/foaf.rdf
7	http://www.cs.umbc.edu/~finin//foaf.rdf
8	http://www.csee.umbc.edu/~finin/finin.rdf
9	http://www-2.cs.cmu.edu/People/fgandon/foaf.rdf
10	http://www.cs.umbc.edu/~kolari1/kolari-foaf.rdf
11	http://www.cs.umbc.edu/~finin//finin.rdf
12	http://www.cs.umbc.edu/~dingli1/foaf.rdf
13	http://lsdis.cs.uga.edu/~amit/foaf.rdf
14	http://trust.mindswap.org/trustFiles/157.owl

Details about this person		no. of source docs
rdfs:seeAlso	http://www.cs.umbc.edu/~finin/finin.rdf	3
rdfs:seeAlso	http://www.cs.umbc.edu/~finin/foaf.rdf	2
rdfs:seeAlso	http://umbc.edu/~finin/foaf.rdf	1
foaf:aimChatID	bimFinin	5
foaf:birthDate	1949-08-04	5
foaf:depiction	http://umbc.edu/~finin/passport.gif	5
foaf:firstName	Tim	5
foaf:homepage	http://umbc.edu/~finin/	5
foaf:mbox	mailto:finin@cs.umbc.edu	7
foaf:mbox	mailto:finin@csee.umbc.edu	5
foaf:mbox	mailto:finin@umbc.edu	5
foaf:mbox_sha1sum	9da08e2b4dc670d9254ab4a4b4d61637fed3b18f	9
foaf:mbox_sha1sum	49953f47b9c33484a753eaf14102af56c0148d37	8
foaf:mbox_sha1sum	8b4d969b2d7dbe0fe5bfc4e069cc2c8a33cf16f4	5
foaf:myersBriggs	ENTP	5
foaf:name	Tim Finin	12
foaf:name	Timothy W. Finin	1
foaf:nick	Tim	5
foaf:phone	tel:+1-410-455-3522	5
foaf:plan	http://www.cs.umbc.edu/~finin/schedule.html	5
foaf:publications	http://www.cs.umbc.edu/%7Efinin/cv/index.shtml#publications	5
foaf:schoolHomepage	http://web.mit.edu/	5
foaf:surname	Finin	5
foaf:weblog	http://ebiquity.umbc.edu/v2.1/blogger/	5
foaf:workInfoHomepage	http://umbc.edu/~finin/	5
foaf:workplaceHomepage	http://umbc.edu/	5

Figure 4-6. Fusing Tim Finin's person profile. For each statement (i.e. triple), the number of supporting sources listed to the right. All supporting sources are listed on left column

The *extensibility* feature of FOAF ontology is exhibited in an *open source* manner – the latest FOAF specification only lists one stable term – 'homepage' and leaves many others in 'testing' or 'unstable' stages. FOAF ontology uses WordNet ontology to define its concepts such as foaf:Agent, and foaf:Person has been used by many other ontology to restrict the *rdfs:domain*. This feature differs from distributed databases in that it preserves a common schema and allows additional customized schema provided by different publishers.

The *inferenceability* feature of FOAF ontology is supported by part of OWL ontology language. Some FOAF properties, such as *foaf:mbox* and *foaf:homepage*, are defined as instance of *owl:InverseFunctionalProperty* and linked the unique identifier of a person.

5.3 Description Logic Reasoning for Adaptive Sensors

A wireless sensor network consists of energy-constrained nodes that are capable of sensing their surrounding environment and communicating that information to a central entity in the network for further processing. Such networks face many challenges including energy management, network management, data management and security [104]. An important and open research problem that combines the above challenges is to design a wireless sensor network that can determine its current state (in terms of energy, communications, security and sensor accuracy) and modify its behavior, if required, to ensure continuous operation as desired by its users.

5.3.1 Sensor State Ontology

One solution to this problem is provided in [105]. In their framework for adaptiveness, the key mechanism is to describe the state of a sensor node and specifies actions that a node should take in each state by a comprehensive OWL ontology that views a sensor node as having multiple components. The ontology describes feasible states of each component and feasible combinations of those states which reflect feasible states associated with a sensor node. The state of each component depends upon certain parameters associated with it. When each parameter takes a specific value, the component is in a particular state. Thus, by defining feasible values for each parameter, we arrive at feasible states of the component.

The hierarchical design of the ontology places the *SensorNode* class at its root. *SensorNode* has properties that describe it in terms of its various components: *Energy*, *PHY*, *MAC*, *Routing*, *Neighborhood* and *Sensor*. Figure 4-7 shows a snippet of the OWL-DL ontology describing its top-level hierarchy. The first two properties only are shown, due to lack of space. The other properties are defined in a similar manner.

```
<owl:Class rdf:ID="SensorNode“/>
<owl:ObjectProperty rdf:ID="hasEnergyComponent">
      <rdfs:domain rdf:resource="#SensorNode"/>
      <rdfs:range rdf:resource="#Energy"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPHYComponent">
      <rdfs:domain rdf:resource="#SensorNode"/>
      <rdfs:range rdf:resource="#PHY"/>
</owl:ObjectProperty>
```

Figure 4-7. Top-level Hierarchy of SensorNode Ontology

5.3.2 Reasoning about Sensor State Using Subsumption

In order to show how the ontology enables a sensor node to determine its state, the energy component of the sensor node requires further investigation. The energy component is defined as shown in Figure 4-8. The state of the energy component depends upon two parameters: *remainingEnergyCapacity* and *energyConsumptionRate*. The former takes its value from a class called *Amount*, which consists of two symbols *Amount_Normal* and *Amount_ Abnormal*. Similarly, the second parameter takes its values from the *Rate*

class. The task of mapping numerical values of these two parameters to logical symbols is performed in a separate module of framework [105] and is beyond the scope of this chapter.

```
<owl:Class rdf:ID="Energy" />
<owl:ObjectProperty rdf:ID="remainingEnergyCapacity">
    <rdfs:domain rdf:resource="#Energy"/>
    <rdfs:range rdf:resource="#Amount"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="energyConsumptionRate">
    <rdfs:domain rdf:resource="#Energy"/>
    <rdfs:range rdf:resource="#Rate"/>
</owl:ObjectProperty>
```

Figure 4-8. Energy Component Definition

Figure 4-9 shows the definition of a *LowEnergyState* associated with a sensor node. A sensor node is in a low-energy state if the remaining energy capacity has an abnormal value *and* the rate of energy consumption is normal. *LowEnergyState* is an intersection of three classes: *Energy* and two anonymous classes that assign particular values to the two parameters. Thus, *LowEnergyState* is an implicit sub-class of *Energy*.

```
<owl:Class rdf:ID="LowEnergyState">
  <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Energy"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#remainingEnergyCapacity"/>
        <owl:hasValue rdf:resource="#Amount_Abnormal"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#energyConsumptionRate"/>
        <owl:hasValue rdf:resource="#Rate_Normal"/>
      </owl:Restriction>
  </owl:I ntersectionOf>
</owl:Class>
```

Figure 4-9. Low Energy State Definition

Figure 4-10 shows how a sensor node in a low energy state is defined in the ontology. *SensorNodeInLowEnergyState* is defined as a sub-class of

SensorNode and has the property that its *Energy* Component is in a low energy state.

```
<owl:Class rdf:ID="SensorNodeInLowEnergyState">
    <rdfs:subClassOf rdf:resource="#SensorNode"/>
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasEnergyComponent"/>
            <owl:someValuesFrom rdf:resource="#LowEnergyState"/>
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

Figure 4-10. Sensor Node in Low Energy State

The rest of the *SensorNode* ontology is designed along similar lines and thus, comprehensively describes feasible states associated with a sensor node. By asserting the ontology into a forward-chaining or backward-chaining reasoning engine that supports OWL-DL, e.g., Java Theorem Prover (JTP), a system that can reason over a given *instance* of a sensor node and respond with a value that indicates the state of the node is created. Thus, in the current example, when the complete instance of a sensor node is presented to a JTP instance containing the ontology, it responds with the correct state of the node. Figure 4-11 shows a snippet of a sensor node instance; only the Energy component is shown due to lack of space.

```
<wsn:SensorNode rdf:ID="SN175">
    <wsn:hasEnergyComponent rdf:resource="#E_SN175"/>
    <wsn:hasPHYComponent rdf:resource="#PHY_SN175"/>
    <wsn:hasMACComponent rdf:resource="#MAC_SN175"/>
    <wsn:hasRoutingComponent rdf:resource="#Rt_SN175"/>
    <wsn:hasNeighborhoodComponent rdf:resource="#Nb_SN175"/>
</wsn:SensorNode>
<wsn:Energy rdf:ID="E_SN175">
    <wsn:remainingEnergyCapacity rdf:resource="#Amount_Abnormal"/>
    <wsn:energyConsumptionRate rdf:resource="#Rate_Normal"/>
</wsn:Energy>
```

Figure 4-11. Complete Instance of a Sensor Node

The response from JTP is shown in Figure 4-12; again only the relevant responses are shown. The reasoning process is automated by designing it as a client-server system. The JTP engine containing the *SensorNode* ontology runs as the server (either on each node or centrally), waits for and receives a sensor node instance, reasons over the instance, responds to the "client" and deletes the current instance from the system. This process continues as long as the node functions.

```
> ask
Enter query: (rdf:type dawsn_v4_embed.owl:SN175 ?x)

Query succeeded.

Bindings 4:
  ?x = |#|::|SensorNodeInLowEnergyState|
Bindings 5:
  ?x = |#|::|SensorNode|
```

Figure 4-12. Query and Response in JTP

5.3.3 Evaluation

The *inferenceability* feature of ontology has been highlighted in this application through the above subsumption inference over sensor network state. The use of OWL-DL inference also makes inference procedure decidable, which is very important to those intelligent but constrained devices which have limited computing capabilities.

6. CONCLUSION

This chapter provides a pragmatic view on Semantic Web ontology via comparison, survey and case-study. The comparison on chronological evolution and built-in semantics reveals how the Semantic Web inherits the merits from pre-existing knowledge representation formalisms and reference models of databases. It also highlights the advantages of using Semantic Web ontology in terms of better visibility, extensibility and inferencability. The practical aspects of Semantic Web ontology is demonstrated through both (i) the significant amount of Semantic Web ontology and instances, and (ii) the rich set of enabling tools for managing and applying Semantic Web ontology. Finally, the three case-studies demonstrate how to use Semantic Web ontology in different application contexts with benefit evaluation in each of them.

ACKNOWLEDGMENTS

Partial support for this research was provided by DARPA contract F30602-00-0591 and by NSF awards NSF-ITR-IIS-0326460 and NSF-ITR-IDM-0219649.

REFERENCES

[1] Smith Barry. Ontology: Philosophical and Computational. http://ontology.buffalo.edu/smith/articles/ontologies.htm; 2000.
[2] Quine WVO. On What There Is. Review of Metaphysics 1948; p. 21–38.
[3] Gruber ThomasR. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 1993; 5(2):199–220.
[4] Uschold Mike, Grüninger Michael. Ontologies: principles, methods, and applications. Knowledge Engineering Review 1996; 11(2):93–155.
[5] Sowa John, editor. Principles of Semantic Networks: Explorations in the Representation of Knowledge. San Mateo: Kaufmann; 1991.
[6] Klyne Graham, Carroll JeremyJ. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/; 2004.
[7] Brickley Dan, Guha RV. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/; 2004.
[8] Dean Mike, Schreiber Guus. OWL Web Ontology Language Reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/; 2004.
[9] Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American 2001; 284(5):35–43.
[10] Minsky Marvin. A Framework for Representing Knowledge. MIT. 1974.
[11] Baader Franz, Calvanese Diego, McGuineness Deborah, Nardi Daniele, Patel-Schneider Peter. The Description Logic Handbook. Cambridge University Press; 2003.
[12] Horrocks Ian, Fensel Dieter, Broekstra Jeen, Decker Stefan, Erdmann Michael, Goble Carole, et al. OIL: The Ontology Inference Layer. Vrije Universiteit Amsterdam; 2000.
[13] Horrocks Ian, Patel-Schneider PeterF, van Harmelen Frank. Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web. In: Proceedings of the Eighteenth National Conference on Artificial intelligence. 2002. p. 792–797.
[14] Codd EF. A relational model of data for large shared data banks. Commun ACM 1970; 13(6):377–387.
[15] Sowa John. Semantic Networks. http://www.jfsowa.com/pubs/semnet.htm, 2002.
[16] Gil Yolanda, Ratnakar Varun. Trusting Information Sources One Citizen at a Time. In: Proceedings of International Semantic Web Conference 2002; 2002. p. 162–176.
[17] Golbeck Jennifer, Parsia Bijan, Hendler James. Trust Networks on the Semantic Web. In: Proceedings of Cooperative Intelligent Agents; 2003.
[18] Richardson Matthew, Agrawal Rakesh, Domingos Pedro. Trust Management for the Semantic Web. In: Proceedings of the 2nd International Semantic Web Conference; 2003.
[19] da Silva PauloPinheiro, McGuinness DeborahL, McCool Rob. Knowledge Provenance Infrastructure. Data Engineering Bulletin 2003; 26(4):26–32.
[20] Carroll JeremyJ, Bizer Christian, Hayes Patrick, Stickler Patrick. Named Graphs, Provenance and Trust. HP Lab; 2004.

[21] Ding Li, Kolari Pranam, Finin Tim, Joshi Anupam, Peng Yun, Yesha Yelena. On Homeland Security and the Semantic Web: A Provenance and Trust Aware Inference Framework. In: Proceedings of the AAAI Spring Symposium on AI Technologies for Homeland Security; 2005.

[22] Ding Zhongli, Peng Yun. A Probabilistic Extension to Ontology Language OWL. In: Proceedings of the 37th Hawaii International Conference On System Sciences, 2004.

[23] Ding Zhongli, Peng Yun, Pan Rong. A Bayesian Approach to Uncertainty Modelling in OWL Ontology. In: Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004). 2004.

[24] Lassila Ora, McGuinness DeborahL. The Role of Frame-Based Representation on the Semantic Web. Stanford University; 2001.

[25] Horrocks Ian, Sattler Ulrike. Description Logics Basics, Applications, and More. Tutorial at ECAI-2002, http://www.cs.man.ac.uk/~horrocks/Slides/ecai-handout.pdf; 2002.

[26] Noy NatalyaFridman, Hafner CaroleD. The State of the Art in Ontology Design: A Survey and Comparative Review. AI Magazine 1997; 18(3):53–74.

[27] Chandrasekaran B, Josephson JohnR, Benjamins VRichard. What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems 1999; 14(1):20–26.

[28] Kagal Lalana, Finin Tim, Joshi Anupam. A Policy Based Approach to Security for the Semantic Web. In: Proceedings of the 2nd International Semantic Web Conference; 2003.

[29] Lenat DouglasB, Guha RV. Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project. Addison-Wesley; 1989.

[30] Miller George. Wordnet: A Dictionary Browser. In: Proceedings of the First International Conference on Information in Data; 1985.

[31] Fellbaum Christiane, WordNet:An Electronic Lexical Database. MIT Press; 1998.

[32] Nirenburg Sergei, Raskin Victor. Ontological Semantics. MIT Press; 2001.

[33] Niles Ian, Pease Adam. Towards a standard upper ontology. In Proceedings of the international conference on Formal Ontology in Information Systems. 2001. p. 2–9.

[34] Beltran-Ferruz PJ, Gonzalez-Calero PA, Gervas P. Converting Frames into OWL: Preparing Mikrokosmos for Linguistic Creativity. In: Workshop on Language Resources for Linguistic Creativity; 2004.

[35] Java Akshay, Finin Tim, Nirenburg Sergei. Integrating Language Understanding Agents Into the Semantic Web. In: Proceedings of AAAI Fall Symposium Series; 2005.

[36] Bench-Capon TrevorJM, Visser PepijnRS. Ontologies in legal information systems; the need for explicit specifications of domain conceptualisations. In: ICAIL-97: Proceedings of the sixth international conference on Artificial intelligence and law. 1997. p. 132–141.

[37] Smith Barry, Williams Jennifer, Schulze-Kremer SchulzeKremer. The Ontology of the Gene Ontology. In: Symposium of the American Medical Informatics Association; 2003.

[38] Lopez MarianoFernandez, Gomez-Perez Asuncion, Sierra JuanPazos, Sierra AlejandroPazos. Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems 1999; 14(1):37–46.

[39] Sklyar Nataliya. Survey of existing Bio-ontologies. Univ. of Leipzig; 2001.

[40] Fonseca FredericoT, Egenhofer MaxJ. Ontology-Driven Geographic Information Systems. In: ACM-GIS; 1999. p. 14–19.

[41] Denny Michael. Ontology Tools Survey, Revisited. http://www.xml.com/pub/a/2004/07/14/onto.html; 2004.

[42] Gennari JohnH, Musen MarkA, Fergerson RayW, Grosso WilliamE, Crubezy Monica, Eriksson Henrik, et al. The evolution of Protege: an environment for knowledge-based systems development. Int J Hum-Comput Stud 2003; 58(1):89–123.
[43] Kalyanpur A., Parsia B., Hendler J.. A Tool for Working with Web Ontologies. In: International Journal on Semantic Web and Information Systems. vol. 1; 2005.
[44] Fikes R, Farquhar A. Large-Scale Repositories of Highly Expressive Reusable Knowledge. IEEE Intelligent Systems 1999; 14(2).
[45] Parsia Bijan, Sirin Evren, Kalyanpu Aditya. Debugging OWL Ontologies. In: In the 14th International World Wide Web Conference (WWW-05); 2005.
[46] Kopena Joseph, Regli William. DAMLJessKB: A Tool for Reasoning with the Semantic Web. IEEE Intelligent Systems 2003; 18(3):74–77.
[47] Fikes Richard, Jenkins Jessica, Frank Gleb. JTP: A System Architecture and Component Library for Hybrid Reasoning. Stanford University; 2003.
[48] Carroll JeremyJ, Dickinson Ian, Dollin Chris, Reynolds Dave, Seaborne Andy, Wilkinson Kevin. Jena: implementing the semantic web recommendations. In: WWW (Alternate Track Papers & Posters); 2004. p. 74–83.
[49] Zou Youyong, Finin Tim, Chen Harry. F-OWL: an Inference Engine for the Semantic Web . In: Formal Approaches to Agent-Based Systems. vol. 3228 of Lecture Notes in Computer Science. Springer-verlag; 2004.
[50] Tsarkov Dmitry, Horrocks Ian. Implementing new reasoner with datatypes support. WonderWeb:Ontology Infrastructure for the Semantic Web Deliverable; 2003.
[51] Horrocks Ian. The FaCT System. In: Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux-98. 1998. p. 307–312.
[52] Haarslev Volker, Moller Ralf. Description of the RACER System and its Applications. In: Proceedings of the International Workshop in Description Logics (DL2001); 2001.
[53] Sirin Evren, Parsia Bijan. Pellet: An OWL DL Reasoner. In: Description Logics; 2004.
[54] Sintek Michael, Decker Stefan. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In: Proceedings of the 1st International Semantic Web Conference (ISWC-02). Springer-Verlag; 2002. p. 364–378.
[55] Harris Stephen, Gibbins Nicholas. 3store: Efficient Bulk RDF Storage. In: Proceedings of the First International Workshop on Practical and Scalable Semantic Systems; 2003.
[56] Lee Ryan. Scalability Report on Triple Store Applications. http://simile.mit.edu/ reports/stores/; 2004.
[57] Guo Yuanbo, Pan Zhengxiang, Heflin Jeff. An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: International Semantic Web Conference; 2004. p. 274–288.
[58] Beckett D., Grant J. Semantic Web Scalability and Storage: Mapping Semantic Web Data with RDBMSes. http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_ mapping_report/; 2003.
[59] Wilkinson K, Sayers C, Kuno H, Reynolds D. Efficient RDF Storage and Retrieval in Jena2. In: Proc. of the 1st International Workshop on Semantic Web and Databases; 2003.
[60] Alexaki Sofia, Christophides Vassilis, Karvounarakis Gregory, Plexousakis Dimitris, Tolle Karsten. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In: Proc. of the 2nd International Workshop on the Semantic Web; 2001.
[61] Broekstra Jeen, Kampman Arjohn, van Harmelen Frank. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Proceedings of the 1st International Semantic Web Conference (ISWC-02); 2002. p. 54–68.
[62] Horrocks Ian, Li Lei, Turi Daniele, Bechhofer Sean. The Instance Store: DL Reasoning with Large Numbers of Individuals. In: Description Logics; 2004.

[63] Pan Zhengxiang, Heflin Jeff. DLDB: Extending Relational Databases to Support Semantic Web Queries. In: Proceedings of the First International Workshop on Practical and Scalable Semantic Systems (PSSS); 2003.

[64] Haase Peter, Broekstra Jeen, Eberhart Andreas, Volz Raphael. A Comparison of RDF Query Languages. In: International Semantic Web Conference; 2004. p. 502–517.

[65] Prud'hommeaux Eric, Seaborne Andy. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/; 2004. W3C Working Draft 12 October 2004.

[66] Stuckenschmidt Heiner, Vdovjak Richard, Broekstra Jeen, Houben GeertJan. Towards Distributed Processing of RDF Path Queries. In International Journal on Web Engineering and Technology; 2005

[67] Cai Min, Frank Martin. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In: WWW-04: Proceedings of the 13th international conference on World Wide Web. ACM Press; 2004. p. 650–657.

[68] Harth A., Decker S. Yet Another RDF Store: Perfect Index Structures for Storing Semantic Web Data With Contexts. http://sw.deri.org/2004/06/yars/doc/summary; 2004.

[69] Stumme G., Maedche A. Ontology Merging for Federated Ontologies on the Semantic Web. In: Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001); 2001.

[70] Stumme G., Maedche A. FCA-Merge: Bottom-up Merging of Ontologies. In: Proceedings of 7th Intl. Conf. on Artificial Intelligence (IJCAI-01); 2001. p. 225–230.

[71] Noy N.F., Musen M.A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000); 2000.

[72] McGuinness D.L., Fikes R., Rice J., Wilder S. An Environment for Merging and Testing Large Ontologies. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR-00); 2000. .

[73] Hovy E. Combining and Standardizing Large-Scale, Practical Ontologies for Machine Translation and Other Uses. In: Proc. of 1st Intl. Conf. on Language Resources and Evaluation; 1998.

[74] Guarino Nicola. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In: SCIE-97: International Summer School on Information Extraction. London, UK: Springer-Verlag; 1997. p. 139–170.

[75] Kiryakov A., Simov K.I. Ontologically Supported Semantic Matching. In: Proceedings of NoDaLiDa-99 Conference; 1999.

[76] Lacher Martin S., Groh Georg. Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. In: Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference. AAAI Press; 2001. p. 305–309.

[77] Prasad S., Peng Y., Finin T. A Tool For Mapping Between Two Ontologies Using Explicit Information. In: AAMAS-02 Workshop on Ontologies and Agent Systems, 2002.

[78] Doan A.H., Madhavan J., Domingos P., Halevy A. Learning to Map between Ontologies on the Semantic Web. In: WWW 2002; 2002.

[79] Doan AnHai, Madhavan Jayant, Dhamankar Robin, Domingos Pedro, Halevy Alon. Learning to match ontologies on the Semantic Web. The VLDB Journal 2003; 12(4):303–319.

[80] Doan A.H., Madhavan J., Domingos P., Halevy A. Ontology Matching: A Machine Learning Approach; 2004. p. 397–416.

[81] Mitra Prasenjit, Wiederhold Gio, Kersten MartinL. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In: EDBT-00: Proceedings of the 7th International Conference on Extending Database Technology; 2000. p. 86–100.

[82] Kalfoglou Y., Schorlemmer M. Information Flow Based Ontology Mapping. In: Proceedings of 1st International Conference on Ontologies, Databases and Applications of Semantics (ODBASE-02); 2002.

[83] Melnik S., Molina-Garcia H., Rahm E. Similarity Flooding: A Versatile Graph Matching Algorithm. In: Proceedings of the Intl. Conf. on Data Engineering (ICDE); 2002.

[84] Noy N.F., Musen M.A. Anchor-PROMPT: Using Non-local Context for Semantic Matching. In: Workshop on Ontologies and Information Sharing at IJCAI-2001; 2001.

[85] Noy N.F., Musen M.A. PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In: AAAI-2002, Edmonton, Canada; 2002.

[86] Noy N.F. Semantic Integration: A Survey Of Ontology-Based Approaches. SIGMOD Record, Special Issue on Semantic Integration 2004; 33(4).

[87] Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufman; 1988.

[88] Mitra P., Noy N.F., Jaiswal A.R. OMEN: A Probabilistic Ontology Mapping Tool. In: Workshop on Meaning Coordination and Negotiation at ISWC-04; 2004.

[89] Ding Z., Peng Y., Pan R., Yu Y. A Bayesian Methodology towards Automatic Ontology Mapping. In: First international workshop on Contexts and Ontologies: Theory, Practice and Applications, held in AAAI-05; 2005.

[90] Pan R., Ding Z., Yu Y., Peng Y. A Bayesian Network Approach to Ontology Mapping. In: Proceedings of the 4th International Semantic Web Conference (ISWC-05); 2005.

[91] Chalupsky H. Ontomorph: A Translation System for Symbolic Knowledge. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR-00). Morgan Kaufman; 2000. p. 471–482.

[92] Dou D., McDermott D., Qi P. Ontology Translation by Ontology Merging and Automated Reasoning. In: Proc. of EKAW Workshop on Ontologies for Multi-Agent Systems; 2002.

[93] Wiesman F., Roos N., Vogt P. Automatic Ontology Mapping for Agent Communication. MERIT; 2001.

[94] Bailin S., Truszkowski W. Ontology Negotiation between Agents Supporting Intelligent Information Management. In: Workshop on Ontologies in Agent-based Systems; 2001.

[95] Bailin S., Truszkowski W. Ontology Negotiation as a Basis for Opportunistic Cooperation between Intelligent Information Agents. In: Proceedings of the Fifth International Workshop on Cooperative Information Agents (CIA 2001); 2001.

[96] Peng Y., Zou Y., Luan X., Ivezic N., Gruninger M., Jones A. Semantic Resolution for E-Commerce. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-02); 2002.

[97] Ciociou M., Nau D. Ontology-Based Semantics. In: Proceedings of the Seventh Intl. Conf. on Knowledge Representation and Reasoning; 2000. p. 539–560.

[98] Stuckenschmidt H., Visser U. Semantic Translation Based on Approximate Re-classification. In: Workshop on Semantic Approximation, Granularity and Vagueness; 2000.

[99] Avancha Sasikanth, Joshi Anupam, Finin Timothy. Enhanced Service Discovery in Bluetooth. Computer 2002; 35(6):96–99.

[100] Farnsworth Dale. Service Discovery Protocol (in Bluetooth Specification). http://www.bluetooth.com/link/spec/bluetooth_e.pdf; 2001.

[101] Avancha Sasikanth. Enhanced Service Discovery in Bluetooth. Master's thesis. UMBC; 2002.
[102] Ding Li, Finin Tim, Joshi Anupam, Pan Rong, Cost RScott, Peng Yun, et al. Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management; 2004.
[103] Ding Li, Zhou Lina, Finin Tim, Joshi Anupam. How the Semantic Web is Being Used:An Analysis of FOAF. In: Proceedings of the 38th International Conference on System Sciences; 2005.
[104] Raghavendra Cauligi S., Sivalingam Krishna M., Znati Taieb, editors. Wireless Sensor Networks. Kluwer Academic Publishers; 2004.
[105] Avancha S., Joshi A., Pinkston J. SWANS: A Framework for Secure, Adaptive Wireless Sensor Networks. UMBC; 2005.