F-OWL: an Inference Engine for the Semantic Web [1]

Youyong Zou, Tim Finin and Harry Chen

Computer Science and Electrical Engineering
University of Maryland, Baltimore County
1000 Hilltop Circle, Baltimore MD 21250
{yzou1,finin, hchen4 }@cs.umbc.edu

Abstract. Understanding and using the data and knowledge encoded in semantic web documents requires an inference engine. F-OWL is an inference engine for the semantic web language OWL language based on F-logic, an approach to defining frame-based systems in logic. F-OWL is implemented using XSB and Flora-2 and takes full advantage of their features. We describe how F-OWL computes ontology entailment and compare it with other description logic based approaches. We also describe TAGA, a trading agent environment that we have used as a test bed for F-OWL and to explore how multiagent systems can use semantic web concepts and technology.

1 Introduction

The central idea of the Semantic Web [Berners-Lee 2001] is to publish documents on the World Wide Web defined and linked in a way that make them both human readable and machine understandable. Human readable means documents in the traditional sense which are intended for machine display and human consumption. Machine understandable means that the data has explicitly been prepared for machine reasoning and reuse across various applications. Realizing the semantic web vision requires well defined languages that can model the meaning of information on the Web as well as applications and services to publish, discover, process and annotate information encoded in them. This involves aspects from many areas, including knowledge representation and reasoning, databases, information retrieval, digital libraries, multi-agent systems, natural language processing and machine learning. The Web Ontology Language OWL [Patel-Schneider, 2003] is part of the growing stack of W3C recommendations related to the Semantic Web. OWL has its origins in DAML+OIL [Hendler 2000] and includes a set of three increasingly complex sub-languages: OWL-Lite, OWL-DL and OWL-Full.

OWL has a model-theoretic semantics that provides a formal meaning for OWL ontologies and instance data expressed in them. In addition, to support OWL-Full, a

[1] This work was partially supported by the Defense Advanced Research Projects Agency under contract F30602-97-1-0215 and by the National Science Foundation under award IIS-0242403.

second model-theoretic semantics has been developed as an extension to the RDF's semantics, grounding the meaning of OWL ontologies as RDF graphs. An OWL inference engine's core responsibilities are to adhere to the formal semantics in processing information encoded in OWL, to discover possible inconsistencies in OWL data, and to derive new information from known information. A simple example demonstrates the power of inference: Joe is visiting San Francisco and wants to find an Italian restaurant in his vicinity. His wireless PDA tries to satisfy his desire by searching for a thing of type *restaurant* with a *cuisineType* property with the value *Italian*. The goodPizza restaurant advertises its cuisine type as *Pizza*. These cannot be matched as keywords or even using a thesaurus, since *Italian* and *Pizza* are not equivalent in all contexts. The restaurant ontology makes things clearer: *Pizza rdfs:SubClassOf ItalianCuisine*. By using an inference engine, Joe's PDA can successfully determine that the restaurant goodPizza is what he is looking for. F-OWL, an inference engine for OWL language, is designed to accomplish this task.

In the next section, we outline the functional requirement of the OWL inference engine. Section three describes F-OWL, the OWL inference engine in Frame Logic that we have developed. Section four explained how F-OWL is used in a multi-agent test bed for trading agents. Chapters five and six conclude this paper with a discussion of the work and results and an outline of some potential future research.

2 OWL Engine

An inference engine is needed for the processing of the knowledge encoded in the semantic web language OWL. An OWL inference engine should have following features:

- **Checking ontology consistency.** An OWL concept ontology (e.g., terms defined in the "Tbox") imposes a set of restrictions on the model graph. The OWL inference Engine should check the syntax and usage of the OWL terms and ensure that the OWL instances (e.g., assertions in the "Abox") meet all of the restrictions.
- **Computing entailments.** Entailment, including satisfiability and subsumption, are essential inference tasks for an OWL inference engine.
- **Processing queries.** OWL inference engines need powerful, yet easy-to-use, language to support queries, both from human users (e.g., for debugging) and software components (e.g., for software agents).
- **Reasoning with rules**. Rules can be used to control the inference capability, to describe business contracts, or to express complex constrictions and relations not directly supported by OWL. An OWL inference engine should provide a convenient interface to process rules that involve OWL classes, properties and instance data.
- **Handling XML data types.** XML data types can be used directly in OWL to represent primitive kinds of data types, such as integers, floating point numbers, strings and dates. New complex types can be defined using base types

and other complex types. An OWL inference Engine must be able to test the satisfiability of conjunctions of such constructed data types.

The OWL language is rooted in description logic (DL), a family of knowledge representation languages designed for encoding knowledge about concepts and concept hierarchies. Description Logics are generally given a semantics that make them subsets of first-order logic. Therefore, several different approaches based on those logics have been used to design OWL inference engines:

- **Using a specialized description logic reasoner.** Since OWL is rooted in description logic, it is not surprising that DL reasoners are the most widely used tools for OWL reasoning. DL reasoners are used to specify the terminological hierarchy and support subsumption. It has the advantage of being decidable. Three well-known systems are FaCT [Horrocks, 1999], Racer [Haarslev 2001] and Pellet. They implement different types of description logic. Racer system implements SHIQ(D) using a Tableaux algorithm. It is a complete reasoner for OWL-DL and supports both Tbox and Abox reasoning. The FaCT system implements SHIQ, but only support Tbox reasoning. Pellet implements SHIN(D) and includes a complete OWL-lite consistency checker supporting both Abox and Tbox queries.
- **Using full first order logic (FOL) theorem prover.** OWL statements can be easily translated into FOL, enabling one to use existing FOL automated theorem provers to do the inference. Examples of this approach include Hoolet (using the Vampire [Riazanov, 2003] theorem prover) and Surnia (using Otter theorem prover). In Hoolet, for example, OWL statements are translated into a collection of axioms which is then given to the Vampire theorem prover for reasoning.
- **Using a reasoner designed for a FOL subset.** A fragment of FOL and general logic based inference engine can also be used to design the OWL inference engine. Horn Logic is most-widely used because of its simplicity and availability of tools, including Jena, Jess, Triple and F-OWL (using XSB). Other logics, like higher-order logic in F-OWL (using Flora), can also be used.

As the following sections describe, F-OWL has taken the third approach. An obvious advantage is that many systems have been developed that efficiently reason over expressive subsets of FOL and are easy to understand and use.

3 F-OWL

F-OWL is a reasoning system for RDF and OWL that is implemented using the XSB logic programming system [Sagonas, 1994] and the Flora-2 [Kifer, 1995] [Yang 2000] extension that provides an F-logic frame-based representation layer. We have found that XSB and Flora-2 not only provide a good foundation in which to implement an OWL reasoner but also facilitate the integration of other reasoning mechanisms and applications, such as default reasoning and planners.

XSB is a logic programming system developed at Stony Brook University. In addition to providing all the functionality of Prolog, XSB contains several features not usually found in Logic Programming systems, including tabling, non-stratified negation, higher order constructs, and a flexible preprocessing system. Tabling is useful for recursive query computation, allowing programs to terminate correctly in many cases where Prolog does not. This allows, for example, one to include "if and only if" type rules directly. XSB supports for extensions of normal logic programs through preprocessing libraries including a sophisticated object-oriented interface called Flora-2. Flora-2 is itself a compiler that compiles from a dialect of Frame logic into XSB, taking advantage of the tabling, HiLog [Chen 1995] and well-founded semantics for negation features found in XSB. Flora-2 is implemented as a set of run-time libraries and a compiler that translates a united language of F-logic and HiLog into tabled Prolog code. HiLog is the default syntax that Flora-2 uses to represent function terms and predicates. Flora-2 is a sophisticated object-oriented knowledge base language and application development platform. The programming language supported by Flora-2 is a dialect of F-logic with numerous extensions, which include a natural way to do meta-programming in the style of HiLog and logical updates in the style of Transaction Logic. Flora-2 was designed with extensibility and flexibility in mind, and it provides strong support for modular software design through its unique feature of dynamic modules.

F-OWL is the OWL inference engine that uses a Frame-based System to reason with OWL ontologies. F-OWL is accompanied by a simple OWL importer that reads an OWL ontology from a URI and extracts RDF triples out of the ontology. The extracted RDF triples are converted to format appropriate for F-OWL's frame style and fed into the F-OWL engine. It then uses flora rules defined in flora-2 language to check the consistency of the ontology and extract hidden knowledge via resolution.

A model theory is a formal theory that relates expressions to interpretation. The RDF model theory [Hayes 2003] formalizes the notion of inference in RDF and provides a basis for computing deductive closure of RDF graphs. The semantics of OWL, an extension of RDF semantics, defines bindings, extensions of OWL interpretations that map variables to elements of the domain:

- The vocabulary *V* of the model is composed of a set of **URI**'s.
- *LV* is the set of *literal values* and *XL* is the mapping from the literals to *LV*.
- A *simple interpretation I* of a vocabulary *V* is defined by:
  - A non-empty set *IR* of resources, called the domain or universe of *I*.
  - A mapping *IS* from *V* into *IR*
  - A mapping *IEXT* from *IR* into the power set of *IR X (IR union LV)* i.e. the set of sets of pairs *<x,y>* with *x* in *IR* and *y* in *IR* or *LV*. This mapping defines the properties of the triples. *IEXT(x)* is a set of pairs which identify the arguments for which the property is true, i.e. a binary relational extension, called the *extension* of x.

Informally this means that every **URI**[2] represents a resource that might be a page on the Internet but not necessarily; it might also be a physical object. A property is a relation; this relation is defined by an extension mapping from the property into a set. This set contains pairs where the first element of a pair represents the subject of a triple and the second element represents the object of a triple. With this system of extension mapping the property can be part of its own extension without causing paradoxes.

Take the triple:*goodPizza :cuisineType :Pizza* from the pizza restaurant in the introduction as example. In the set of **URI**'s there will be terms (i.e., classes and properties) like: *#goodPizza, #cuisineType, #pizza, #Restanrant, #italianCuisine,* etc. These are part of the vocabulary *V*. The set **IR** of resources include instances that represent resources on the internet or elsewhere, like *#goodPizza*, , etc. For example the class *#Restanrant* might represent the set of all restaurants. The **URI** refers to a page on the Internet where the domain **IR** is defined. Then there is the mapping *IEXT* from the property *#cuisineType* to the set {(*#goodPizza, #Pizza),(#goodPizza, #ItalianCuisine)*} and the mapping *IS* from *V* to *IR*: *:goodPizza → #goodPizza, :cuisineTYpe → #cuisineType.*

A rule *A→B* is satisfied by an interpretation *I* if and only if every binding that satisfies the antecedent *A* also satisfies the consequent *B.* An ontology *O* is satisfied by an interpretation *I* if and only if the interpretation satisfies every rules and facts in the ontology. A model is satisfied if none of the statements within contradict each other. An ontology O is consistent if and only if it is satisfied by at least one interpretation. An ontology O_2 is entailed by an ontology O_1 if and only if every interpretation that satisfies O_1 also satisfies O_2.

One of the main problems in OWL reasoning is ontology entailment. Many OWL reasoning engines, such as Pellet and SHOQ, follow an approach suggested by Ian Horrocks [Horrocks 2003]. By taking advantage of the close similarly between OWL and description logic, the OWL entailment can be reduced to knowledge base satisfiability in the SHOIN(D) and SHIF(D). Consequently, existing mature DL reasoning engines such as Racer [Haarslev 2001] can provide reasoning services to OWL. Ora Lassila suggested a *"True RDF processor"* [Lassila 2002] in his implementation of Wilbur system [Lassila 2001] in which entailment is defined via the generation of a deductive closure from an RDF graph composed of triples. The proving of entailment becomes the building and searching of closure graph.

With the support of forward/backward reasoning from XSB and frame logic from Flora, F-OWL takes the second approach to compute the deductive closure of a set of RDF or OWL statements. The closure is a graph consisting of every triples <*subject, predicate, object*> that satisfies *{subject, object } ⇒ IEXT(I(predicate)).* This is defined as:

$$<subject,predicate,object> \Rightarrow KB \Leftrightarrow \{subject,object\} \Rightarrow IEXT(I(predicate))$$

[2] The W3C says of URIs: "Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources." By convention, people understand many URIs as denoting objects in the physical world.

Where *KB* is the knowledge base, *I(x)* is the interpretation of a particular graph, and *IEXT(x)* is the binary relational extension of property as defined in [Hayes 2002].

F-OWL is written in the Flora-2 extension to XSB and consists of the following major sets of rules:

- A set of rules that reasons over the data model of RDF/RDF-S and OWL;
- A set of rules that maps XML DataTypes into XSB terms;
- A set of rules that performs ontology consistency checks; and
- A set of rules that provides an interface between the upper Java API calls to the lower layer Flora-2/XSB rules.

F-OWL provides command line interface, a simple graphical user interface and a Java API to satisfy different requirements. Using F-OWL to reason over the ontology typically consists of the following four steps:

- Loading additional application-related rules into the engine;
- Adding new RDF and OWL statements (e.g., ontologies or assertions) to the engine. The triples (subject, predicate, object) on the OWL statements are translated into 2-ply frame style: subject(predicate, object)@model;
- Querying the engine. The RDF and OWL rules are recursively applied to generate all legal triples. If a query has no variables, a True answer is returned when an interpretation of the question is found. If the question includes variable, the variables is replaced with values from the interpretation and returned;
- The ontology and triples can be removed if desired. Else, the XSB system saves the computed triples in indexed tables, making subsequent queries faster.

4 F-OWL in TAGA

Travel Agent Game in Agentcities (TAGA) [Zou 2003] is a travel market game developed on the foundation of FIPA technology and the Agentcities infrastructure. One of its goals is to explore and demonstrate how agent and semantic web technology can support one another and work together.

TAGA extends and enhances the Trading Agent Competition scenario to work in Agentcities, an open multiagent systems environment of FIPA compliant systems. TAGA makes several contributions: auction services are added to enrich the Agentcities environment, the use of the semantic web languages RDF and OWL improve the interoperability among agents, and the OWL-S ontology is employed to support service registration, discovery and invocation. The FIPA and Agentcities standards for agent communication, infrastructure and services provide an important foundation in building this distributed and open market framework. TAGA is intended as a platform for research in multiagent systems, the semantic web and/or automated trading in dynamic markets as well as a self contained application for teaching and experimentation with these technologies. It is running as a continuous open game at

http://taga.umbc.edu/ and source code is available on Sourceforge for research and teaching purposes.

The agents in TAGA use OWL in various ways in communication using the FIPA agent content language (ACL) and also use OWL-S as the service description language in FIPA's directory facilitators. Many of the agents in the TAGA system use F-OWL directly to represent and reason about content presented in OWL. On receiving an ACL message with content encoded in OWL, a TAGA agent parses the content into triples, which are then loaded into the F-OWL engine for processing.

When an agent receives an incoming ACL message, it computes the meaning of the message from the ACL semantics, the protocols in effect, the content language and the conversational context. The agent's subsequent behavior, both internal (e.g., updating its knowledge base) and external (e.g., generating a response) depends on the correct interpretation of the message's meaning. Thus, a sound and, if possible, complete understanding the *semantics* of the key communication components (i.e., ACL, protocol, ontologies, content language, context) is extremely important. In TAGA, the service providers are independent and autonomous entities, which making it difficult to enforce a design decision that all use exactly the same ontology or protocol. For example, the Delta Airline service agent may have its own view of travel business and uses class and property terms that extend an ontology used in the industry. This situation parallels that for the semantic web as a whole – some amount of diversity is inevitable and must be panned for lest our systems become impossibly brittle.

Many of the agents implemented in TAGA system use F-OWL to represent and reason about the message content presented in RDF or OWL. Upon receiving an ACL message with content in RDF or OWL, a TAGA agent parses the content into triples, which are then loaded into the FOWL engine for processing.

The message's meaning (communicative act, protocol, content language, ontologies and context) all play a part in the interpretation. For example, when an agent receives a query message that uses the query protocol, the agent searches its knowledge base for matching answers and returns an appropriate inform message. TAGA uses multiple models to reflect the multiple namespaces and ontologies used in the system. The agent treats each ontology as an independent model in the F-OWL engine.

F-OWL has many usages in TAGA, including the following.

- **As knowledge base.** Upon receiving an ACL message with content encoded in OWL, agents in TAGA parse the content into triples and feeds them into their F-OWL engine. The information can be easily retrieved by submitting queries in various query languages.
- **As reasoning engine.** The agent can answer more questions with the help of F-OWL engine, for example, the restaurant can answer the question "what is the average price of a starter" after it understands that "starter" is *sameAs* "appetizer".
- **As a service matchmaker.** FIPA platforms provide a *directory facilitator* service which matches service requests against descriptions of registered services. We have extended this model by using OWL-S as a service description language.

F-OWL manages the service profiles and tries to find the best match based on description in the service request.

- **As an agent interaction coordinator.** The interaction protocol can be encoded into an ontology file using OWL language. F-OWL will advise the agents what to respond based on received messages and context.

5 Discussion

This section describes the design and implementation of F-OWL, an inference engine for OWL language. F-OWL uses a Frame-based System to reason with OWL ontologies. F-OWL supports consistency checking of the knowledge base, extracts hidden knowledge via resolution and supports further complex reasoning by importing rules. Based on our experience in using F-OWL in several projects, we found it to be a fully functional inference engine that was relatively easy to use and able to integrate with multiple query languages and rule languages.

There have been lots of works on the OWL inference engine, from semantic web research community and description logic community. The following table compares F-OWL with some of them:

Table 1: Comparison of F-OWL and other OWL Inference Engine

	F-OWL	**Racer**	**FaCT**	**Pellet**	**Hoolet**	**Surnia**	**Triple**
Logic	Horn, Frame, Higher Order	Description Logic	DL	DL	Full FOL	Full FOL	Horn Logic
Support	OWL-Full	OWL-DL	OWL-DL	OWL-DL	OWL-DL	OWL-Full	RDF
Based on	XSB/Flora	Lisp	Lisp	Java	Vampire	Otter	XSB
XML Datatype	Yes	Yes	No	Yes	No	No	No
Decidable	No	Yes	Yes	Yes	No	No	Yes
Complete consistency checker	No	Yes (OWL-Lite)	Yes	Yes(OWL-Lite)	No	No	No
Interface	Java, GUI, Command Line	DIG, Java, GUI	DIG, Command Line	DIG, Java	Java	Python	Java
Query	Frame style, RDQL	Racer query language		RDQL			Horn logic style
Known Limitation	Poor scaling		No Abox support		Poor scaling	Poor scaling	Only support RDF

The first thing to notice in Table 1 is that the description logic based system can only support reasoning over OWL-Lite and OWL-DL statements but not OWL-Full. OWL-Full is a full extension of RDF, which needs the supporting of terminological cycle. For example, a class in OWL-Full can also be an individual or property. The cyclic terminological definitions can be recognized and understood in horn logic or frame logic system.

Table 1 shows that only three DL-based owl inference engines, which are all use a Tableau based algorithms [Baader 2000], are decidable and support complete consistency checking (at least in OWL-Lite). However, [Balaban 1993] argues that DL only forms a subset of F-Logic. The three kinds of formulae in the description logic can be transformed into first class objects and n-ary relationships. F-Logic is able to provide a full account for DL without losing any semantics and descriptive nature. We understand that our current F-OWL approach is neither decidable nor complete. However, a complete F-Logic based OWL-DL reasoner is feasible.

The table also shows that F-OWL system doesn't scale well when dealing with large datasets, because of the incompleteness of the reasoner. Actually, none of the OWL inference engines listed here scales well when dealing with the OWL test case wine ontology[3] which defines thousands of classes and properties and a relatively modest number of individuals. Further research is needed to improve the performance and desirability.

Comparing with other OWL inference engines, F-OWL has several unique features: tabling, support for multiple logical models or reasoning, and a pragmatic orientation.

Tabling. XSB's tabling mechanism gives F-OWL the benefits of a forward chaining system in a backward chaining environment. The triples in a model are computed only when the system needs to know whether or not they are in the model. Once it is established that a triple is in the current model, it is added to the appropriate table, obviating the need to prove that it is in the model again. This mechanism can have a significant impact on the system's performance. While the first few queries may take a long time, subsequent queries tend to be very fast. This is an interesting compromise between a typical forward-only reasoning system and backward-only reasoning systems.

Multiple logics. F-OWL supports Horn logic, frame logic and a kind of higher-order logic; all inherited from the underlying XSB and Flora substrates. Working together, these logic frameworks improve F-OWL's performance and capabilities. For example, the F-logic supports non-monotonic (default) reasoning. Another example is higher-order logic. The semantics of higher-order logics, in general, are difficult and in many cases not suitable for practical applications. XSB's Hilog, however, is a simple syntactic extension of first-order logic in which variables can appear in the position of a predicate. In many cases, this simplifies the expression of the statements, rules and constraints, improving the writability and readability of F-OWL and associated programs.

---

[3] The wine ontology is used as a running example in the W3C's OWL Web Ontology Language Guide and is available at http://www.w3.org/TR/owl-guide/wine.owl.

Pragmatic approach. The aim of F-OWL system is to be a practical OWL reasoner, not necessary a complete OWL reasoner. So F-OWL system provides various interface to access the engine and supports multiple query and rule languages.
In the open web environment, it is generally assumed that the data are not complete and not all facts are known. We will research how this fact affects the implementation of inference engine. In the semantic web an inference engine may not necessarily serve to generate proofs but should be able to check proofs. We will work on using F-OWL to resolve trust and proof in semantic web.

In a stand-alone system inconsistencies are dangerous but can be controlled to a certain degree. However, controlling the inconsistencies in the Semantic Web is a lot more difficult. During the communication, ontology definition origin from other agents, who is unknown beforehand, may be asserted. Therefore special mechanisms are needed to deal with inconsistent and contradictory information in the Semantic Web. There are two steps: detecting the inconsistency and resolving the inconsistency.

The detection of the inconsistency is based on the declaration of inconsistency in the inference engine. The restriction, which imposes the possible values and relation that the ontology elements can have, leads to the inconsistency. For example, *owl:equivalentClass*: imposes a restriction on the resource which the subject is same class as. *owl:disjointWith* imposes a restriction on the resource which the subject is different from. The triples *(a owl:equivalentClass b)* and *(a owl:disjointWith b)* is not directly lead to an inconsistency until applying the detection rule: *(A owl:equivalentClass B) & (A owl:disjointWith B)* ➔*inconsistency.*

When inconsistencies are detected, Namespaces can help tracing the origin of the inconsistencies. John posted "all dogs are human" at his web site, while "all dogs are animal" appears in daml.org's ontology library. It is clear that the second is more trustable. Every web site are identified and treated unequivocally in the semantic web. The inference engine contacts trust system to evaluate the creditability of the namespaces. [Klyne 2002] and [Golbeck 2003] enlist lots of works and brilliant ideas about how to maintain the trust system in the semantic web. Once having the trust evaluation result, the agent could take three different actions: (a) accept the one suggested by the inference engine; (b) reject both as none of them is trustable; (c) ask the human user to select.

6 Conclusion

This paper describes the design and implementation of F-OWL, an inference engine for OWL language. F-OWL uses a Frame-based System to reason with OWL ontologies. F-OWL supports consistency checking, extracts hidden knowledge via resolution and supports further complex reasoning by importing rules. While using it in TAGA user case, we find that F-OWL is a full functional inference engine and easy to use with the support of multiple query languages and rule languages.

In the open web environment, it is generally assumed that the data are not complete and not all facts are known. We will research how this fact affects the implementation of inference engine. In the semantic web an inference engine may not nec-

essarily serve to generate proofs but should be able to check proofs. We will work on using F-OWL to resolve trust and proof in semantic web in the future.

References

[Baader 2000] Franz Baader and Ulrike Sattler: "An Overview of Tableau Algorithms for Description Logics", Proceeding of Tableau 2000, RWTH Achen.

[Balaban 1993] Mira Balaban: "The F-Logic Approach for Description Languages", Ben-Gurion University of Negev Technical Report FC-93-02, 1993.

[Berners-Lee 2001] Tim Berners-Lee, James Hendler and Ora Lassila: "The Semantic Web", Scientific America , May 2001.

[Chen 1995] Weidong Chen, Michael Kifer and David Warren: "Logical Foundations of Object-Oriented and Frame-Based Languages", Journal of ACM, May 1995.

[Golbeck 2003] Jennifer Golbeck, Bijan Parsia, and James Hendler: "Trust networks on the semantic web", Proceedings of Cooperative Intelligent Agents 2003, Helsinki, Finland, August 2003.

[Haarslev 2001] Volker Haarslev and Ralf Moller: "Racer system description", Proceeding of International Joint Conference on Automated Reasoning, Volume 2083, page 701-705, Springer 2001

[Hayes 2003] Pat Hayes, RDF Semantics , W3C working Draft, 2003.

[Hendler 2000] James Hendler and Deborah L. McGuinness, "The DARPA Agent Markup Language." IEEE Intelligent Systems, Trends and Controversies, page. 6-7, November/December 2000.

[Horrocks, 1999] I. Horrocks. FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133-135, 1999.

[Horrocks, 2003] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17-29. Springer, 2003.

[Kifer, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. JACM, 42(4):741--843, Jul. 1995.

[Klyne 2002] G.Klyne: "Framework for Security and Trust Standards", SWAD-Europe, December 2002.

[Lassila 2001] Ora Lassila: Enabling Semantic Web Programming by Integrating RDF and Common lisp", Proceeding of first Semantic Web Working Symposium, Stanford University, 2001.

[Lassila 2002] Ora Lassila: **"**Taking the RDF Model Theory Out for a Spin", First International Semantic Web Conference (ISWC 2002), Sardinia (Italy), June 2002.

[Patel-Schneider 2003] Peter F. Patel-Schneider, Pat Hayes and Ian Horrocks, OWL Web Ontology Language Semantics and Abstract Syntax, W3C working Draft, 2003.

[Riazanov, 2003] A.Riazanov, Implementing an Efficient Theorem Prover, PhD thesis, University of Manchester, 2003.

[Sagonas, 1994] Kostantinos Sagonas, Terrance Swift, and David S. Warren: XSB as an efficient deductive database engine, In ACM Conference on Management of Data (SIGMOD), 1994.

[Yang, 2000] Guizhen Yang and Michael Kifer. FLORA: Implementing an efficient DOOD system using a tabling logic engine. Proceedings of Computational Logic --- CL-2000, number 1861 in LNAI, pp 1078--1093. Springer, July 2000.

[Zou, 2003] Youyong Zou, Tim Finin, Li Ding, Harry Chen, and Rong Pan, TAGA: Trading Agent Competition in Agentcities, Workshop on Trading Agent Design and Analysis, held in conjunction with the Eighteenth International Joint Conference on Artificial Intelligence, Monday, 11 August, 2003, Acapulco MX.

[Zou, 2004] Youyong Zou, Agent-Based Services for the Semantic Web, Ph.D. Dissertation, University of Maryland, Baltimore County, August, 2004.