M.S. Essay

Computational Semantics and Type Theory and its influence on Semantic Web and Related Applications – An Introduction

Preeda Rajasekaran
(Under the Direction of John A. Miller)

LSDIS Lab, Computer Science Department, University of Georgia, Athens, 30602

(*preeda@cs.uga.edu*)

1. Introduction

Semantics- the concept of 'study of meaning' is changing the way content is currently organized in the World Wide Web. The Web today gives us unlimited access to a plethora of content, but this information cannot be processed to its full potential as a 'source of knowledge' due to the lack of means to 'understand' the underlying data. The next-generation of the Web, the Semantic Web, overcomes this obstacle by 'adding meaning to content', thus making the information reclaimable. This facilitates heterogeneous, distributed information to be used by any user who has access to the content, to tailor it for their respective needs. Acquiring a detailed understanding of semantics, the corner stone of the Semantic Web, will enable us to tap into the applications built on top the Semantic Web framework. This essay gives an insight into the basics of semantics and its application in Semantic Web technologies such as Semantic Web Services (SWS). The importance of SWS has been marked by its participation in real-world applications such as EAI (Enterprise Application Integration) and Electronic Commerce. Both these fields involve knowledge management, and are moving towards the ultimate goal of 'anyone can trade with anyone else' principle. This demands the capability for dealing with numerous heterogeneous data types. Understanding the semantics of types plays an important role in bringing about seamless application integration. A short discussion on type theory and strongly-typed languages (functional languages) like Haskell is presented in this essay to bring about deeper understanding of types and their contribution in enabling Semantic Web applications. METEOR-S (Managing End-To-End OpeRations for semantic Web Services) [METEOR-S, 2002] and OWL-S (Web Ontology Language-Services) [OWL-S, 2003] are research initiatives taken in the direction of realizing the Semantic Web. An overview of the approaches followed by these systems to harness the power of semantics is also discussed in this essay.

2. Semantics

The word *semantics* is derived form the Greek word *sematikos* –which means significant meaning, derived from *sema* sign. Semantics, which deals with meaning/content, is one of the three branches of semiotics. The other two branches are syntax and pragmatics. Syntax is defined as 'the study of formal relationships between words' [Jurafsky & Martin, 2000]. Pragmatics, deals with 'study of the relation between language and context of use' [Jurafsky & Martin, 2000]. Semantics differs from syntax (the formal structure or pattern) in pertaining to what something means. Semantics the 'study of meaning', can also be explained as the relationship between word symbols and their intended meaning. Formal logic semantics is important as it helps in defining a language in a methodical way. The formal representation of semantics helps to perform querying and reasoning.

The different types of semantics are:

i) Denotational Semantics

Denotational meaning can be expressed as 'Knowing that', which defines meaning, as reference. In logic theory denotational semantics formalizes the meaning of languages/programs by means of mathematical functions.

ii) Operational Semantics

Operational meaning can be defined as 'Knowing How'. It is the valid algorithm for performing actions. Operational semantics is used to understand a language via of a state transition system for the language under consideration. This is enabled by defining a set of formal inference rules to define the valid transitions of the system.

The difference between denotational and operation semantics can be explained by the following example:

5 + 3 = 8 (addition)
4 * 2 = 8 (multiplication)

Both results give the natural number 8. The denotational meaning of both the statements is the same, i.e. the result- natural number 8, but the operational meaning of the statements 'addition' and 'multiplication' respectively is different.

iii) Axiomatic semantics

Axiomatic semantics is used to define assertions about properties of a system. It is based on mathematical logic to prove the correctness of the system. It describes how the state change of the system affects these assertions. In terms of programming, assertions are constraints such as pre/post-conditions and invariants of an operation, and axiomatic semantics deals with the state of these constraints after the execution of the operation.

iv) Model-Theoretic Semantics:

Model-Theoretic Semantics [Farrugia, 2003] deals with the study of language meaning using mathematical and logical formalism. Semantics/Meaning of language is given by the interpretation of mathematical concepts of set-theory; this enables explaining the semantics via formal structures such as predicates. Formalism helps to bring about automated reasoning. Inferencing and reasoning are based on resolution theory. Resolution-theorem provers [Nivelle, 2000] are based on the principle of- 'proof by contradiction'. Predicates of a formalized language are used as inputs to theorem provers to deduce their truth value and thus can be used in perform inferencing and reasoning.

3. Need for Semantics

The current Web can be viewed as a collection of documents (static and dynamic) made up of word symbols. While as a third-party we have access to this information, automating the interpretation and utilization of the knowledge present in them to suit our own purpose is currently not possible. This is because the 'meaning' of the data content, as thought of by the publisher of the document is not available for others explicitly. Humans can make use their intuition to make sense of the documents and process them accordingly, but the absence of machine processable information to describe the content is a huge hindrance to automating the management of knowledge present in the Web. The

machine processable content that gives meaning to the data published is referred to as meta-data or data semantics.

Semantics provides 'interpretations' of formal languages. This formalism of semantics helps us to express the meaning of content, depending on the context of use, in a machine processable and interpretable manner. 'Understanding' the content of the data published on the Web not only helps in automating a number of tasks, but also serves to improve the efficiency of searching, filtering and categorizing information on the Web.

The meaning of content varies depending upon the context of use and the intention of the publisher. For the published content to be successfully used by others, in the right sense, semantics offers valuable assistance in deciphering the information. E-Business today is highly dynamic with ever-changing business needs, varying partners and heterogeneous content management. Semantic meta-data helps to make these business transitions less time consuming, more efficient and with increased ROI (Return Of Investment). The reason being agreements with partners can be reached automatically as the application semantics helps in data-integration, exchange of policies and agreements. Moreover, the business logic can be unambiguously defined with the help of semantics to be understood by different partners.

4. Semantic Web

The Semantic Web can be defined as 'an extended Web of machine-readable information and automated services that extends far beyond current capabilities' ([Berners-Lee et al., 2001], [Fensel & Musen, 2001]). Adding explicit semantics to underlying content will transform the Web into a global knowledge source that can prove useful to a number of applications. As opposed to the information overload in the current Web, the Semantic Web will help present the content in a more organized manner. The inherent nature of Semantic Web technology (semantically enriched) helps develop new and flexible approaches to Data Integration [Fensel et al., 2002]. This facilitates many applications, particularly Semantic Web Services, which are built using the infrastructure of the Semantic Web [Hawke, 2002].

5. Semantic Web Services

Semantic Web Services combine the advantages of Web Services with the Semantic Web to provide valuable support for information access and e-business. Web Services can be defined as "self-contained, self-describing, modular applications that can be published, located, and invoked across the Web" [IBM Web Service Tutorial]. Web Services provide executable services and they are described using WSDL (Web Service Description Language) files. While these descriptions contain information about the operation and parameter names in the service, they offer little/no information about the functionality of the service. Moreover, the description of services present in UDDI, (repository of services) is not machine processable (informal descriptions). These informal descriptions force human intervention in discovering, composing and invoking Web services. Incorporation of semantics helps to provide a formal representation of informal descriptions. Consider a service offering functionality to calculate Financial Bond Price. Search for this service in the current UDDI would employ the keywords "Bond", "Price", "calculate" and wild-card character "%", as the exact name of the service is unknown to the user. The precise context of search is lost due to the presence of wild card character and the different meanings of the keywords employed in search. For example the usage of Bond has many contexts, such as "Financial Bond", "Bail Bond", "Human Bond" etc. While humans can distinguish among the various usage, search results contain services in which the word "Bond" occurs in any context in the description. By adding semantic

annotations using ontologies (e.g. annotating the method 'calculateBondPrice' with 'Finance:#calculateBond' – a concept in the Finance Ontology), we can provide the context in which a word is used in the real world. Ontology being a representation of the real world, facilitates this. As the correct context of the word is employed in search, we can expect more precise results being returned.

With the support offered by semantic Web services, the functionalities of Web services, such as discovery, composition and invocation can be automated to a great extent. METEOR-S, OWL-S and DERI are research initiatives in this direction. Their approaches to adding semantics for Web services development will be discussed later in the essay. Knowledge is represented in the semantic Web mainly via Ontologies. Ontologies serve as the main source of semantics information for the content present in Semantic Web.

6. Ontology

The core of the semantic knowledge present in the Semantic Web is through Ontologies. Ontology can be aptly described as a 'formal specification of conceptualization shared in a community' [Gruber, 1993]. It can also be looked upon as a vocabulary of terms and relations that is used to represent an unambiguous view of the world. Ontologies help to formalize the communication across the applications and extensions of Semantic Web. The components of ontology can be briefly stated as (i) Concepts (Vocabulary) (ii) Structure (hierarchy of concepts and their attributes) (iii) Specific characteristics of concepts and their attributes (e.g., Domain and Range Restrictions, Properties of relations). A clear perspective on the basics of ontology and its features can help enhance the semantic knowledge resident in Semantic Web.

The Web Ontology Language –OWL [McGuinness & Harmelen, 2004] is a new formal language for representing ontologies in the semantic Web. OWL characteristics are obtained as extensions to RDF (Resource Description Framework) and RDF-S (RDF-Schema). OWL's semantics are grounded in DL expressivity. [Horrocks et. al., 2003]. An ontology can be viewed as a Description Logic knowledge base. Description Logic (DL) is a knowledge representation formalism (KR) that represents a domain. The domain is described in terms the concepts in the domain, properties used to define these concepts and relationship between the concepts in the domain. Using DL as a foundation of ontologies help us use its inherent reasoning to make implicit knowledge explicit. Based on the level of expression and time-complexity of reasoning OWL comes in three flavors,

- OWL-Lite-which uses simple constraints and reasoning, and is computationally simple and efficient
- OWL- DL-which is computationally complete and decidable.
- OWL-Full-which offers the most maximum expressiveness, but offers no computational guarantees.

Ontologies used in METEOR-S support multiple cardinalities, subClassOf and Equivalent Classes relations and other properties, which require the use of OWL–DL. OWL-DL offers sufficient expressiveness to ontologies, and at the same time offers decidable time-complexity for performing subsumption based reasoning.

Ontologies organize entities into classes based on their properties and relationship with other entities. Concepts similar to ontologies occur in a variety of disciplines, they are as follows [Lagoze, 2002]:

- Domains: Database theory
- Types: AI, Compilers, Programming Languages
- Classes: OO systems
- Types/Sorts: Logic

Ontologies facilitate communication between business partners by offering an unambiguous representation of the data interchanged through semantic data/type information. In order to bring about successful implementation of data integration using ontologies, we need to find appropriate mapping to semantic concepts. In some cases type mapping and transformation functions may need to be applied. The motivating factor behind this is that, in a real-world scenario, it is often difficult to find matches between ontological concepts and data represented in applications. Often the mapping is provided to the closest ontological concept, with transformation rules to bring about correlation between the mappings.

Type theory and functional languages provide an insight into the importance of type and fundamentals of type system. The useful properties offered from these two fields can be adapted to enhance the features of type system as offered by ontologies, i.e., to include type inferencing, type transformations, etc. Agreement of types between partners in e-business plays a vital role in bringing about seamless, dynamic process composition, the importance of which increases many-fold when the business environment is dynamic and partners change often.

7. Type Theory, Functional Languages and Ontologies

Type theory [Eijck, 2003] and functional languages have their basic concepts deeply rooted in 'types'. What are types? 'Type' can be defined as 'specified kind'. In English grammar, type corresponds to adjective, which represents 'a word that expresses an attribute of something'. In programming language types refer to data-types such as 'int', 'float' etc. Types in ontologies refer to ontology classes. Ontology classes represent different concept types in a given domain. In general, types are used to disambiguate an entity/thing/concept on the basis of its properties, function and relationships.

Type theory is a branch of mathematics and logic, which uses the notion of type to classify entities into sets called 'types'. A language based on 'type theory' has every expression mapped to a function of appropriate type. In 'type theory',

-Basic types refer to type of expression – which refer to individual objects.

-Complex types refer to classes.

-Properties and Relations are expressed as functions that take in arguments (properties) and return a truth value (boolean).

Application of type theory in programming languages helps define a 'type system' [Eijck, 2003]. A type system helps to classify program values into sets called 'types'. Type system thus enforces restrictions for developing well-formed programs. This feature is essential in detecting compile time errors based on type mismatch. A type system in a way delineates the functionality of a language [Gunter, 1992]. Applying the concepts of 'type system' to ontologies, we can draw an analogy. An entity in an application which is annotated with an ontological concept should satisfy all the properties associated with the corresponding annotated ontology class (concept). If this condition is not satisfied then we need to apply transformation rules to obtain the required mapping. Validators for annotation (similar to compilers for programming languages) should ensure this.

Functional languages such as Haskell [Browne, 2002] are strongly-typed, which is means that all data-types are known during compilation (static typing). While this may sound too restrictive, this ensures almost error free programming, as most errors are caught during compile time. If a similar principle of knowing the semantic types before compilation could be strictly followed in annotating semantic Web services, then dynamic composition of services might be hassle free and seamless. Moreover, adapting functionalities of a type-system similar to Haskell will help develop,

- Type inferencing -For computing missing type annotations.
- Type coercion –For implicit casts generated automatically by the compiler/system.

With regard to the Web Service Description Language, the most commonly used type system is XML Schema, whereas Web Services implementation is done using languages like C# .NET and Java. Though there is direct mapping between primitive data types of languages (Java) and XML schema, the same cannot be said about complex and user defined data-types. They require the service provider to give the appropriate transformations/mapping to XSD (XML Schema Definition) types. As the semantic type of the operation/parameters is specified via OWL standards, we also require transformation rules to map between Java data-types and OWL data-types [Kalyanpur et al., 2004]. Ideally, a complete round-tripping between the three type systems above is desirable to maintain coherence between the annotated description of the Web services published and the semantically enhanced implementation code of the Web Services. Round-tripping between different type-systems would require extensive research on the various type systems involved and has to take into consideration the difference in design and functionality of the each type systems.

8. OWL-S

OWL-S is one of the modeling frameworks for developing Semantic Web Service. It defines an upper ontology of services (central class for describing service interfaces). Semantic Web Services developed using OWL-S [OWL-S, 2004] are considered as the instances of this upper ontology. While the fundamental features for a service are specified in the upper ontology, application specific extensions can be made to the ontology. OWL-S services consists of four files [Paolucci et al., 2002a]

1) Profile (.owl) - Describes the functional (input, output, preconditions and effects) and non-functional aspects of the service.
2) Process (.owl) - Describes the service's operations and the interaction protocol of the service.
3) Grounding (.owl) – Provides mapping from abstract (Process model) to concrete (WSDL) representation.
4) WSDL (.wsdl) file for the service.

The main placeholder of semantic information is the Profile (.owl) file, which contains the annotations for IOPE (Inputs, Outputs, Pre-Conditions and Effects). Preconditions and Effects of an operation are defined using SWRL [Horrocks et al., 2004].(Semantic Web Rule Language). It is the combination of OWL-DL and Unary/Binary Datalog or RuleML [Boley et al., 2004]. The constraints are expressed via Horn-logic rules for OWL language. The inability of OWL to express mathematical expressions necessitates the use of SWRL. SWRL is also used to perform reasoning and inferencing in OWL-S.

Pre-conditions and effects are expressed as logical formulas to facilitate reasoning and inferencing. OWL-S [OWL-S, 2004a] explores different ways of representing these constraints. Constraints can be expressed as string literal or as XML literals. SWRL employs XML literals to represent constraints. Languages such as KIF (Knowledge Interchange Format) [Klyne & Carroll, 1998] and PDDL (Planning Domain Definition Language) [Ghallab, 1998] are used to symbolize constraints as string literals. DRS (**Discourse Representation Structure)** is also being considered as an expression language. An ontology for expressions is also being developed as a part of OWL-S to define expressions, e.g., to define what a non-negative number is in terms of properties and relations in the ontology. This ontology will be used in conjunction with constraints to define pre-conditions and effects, e.g., pre-condition before placing an order, (User BankBalance > 0).

9. DERI

Digital Enterprise Research Institute's mission can be defined as 'to make Intelligent Web Services an reality'. This institute led by Dr. Dieter Fensel and Dr. Christoph Bussler focuses on Semantic Web Services oriented research. The main research groups in DERI [DERI, 2003] oriented towards the area of Semantic Web Services(SWS) are

- Ontology Management Working Group (OMWG)
  – an Ontology Management Suite, consisting of tools to handle versioning, merging, editing and browsing of ontology. It is also provided with an interpreter for querying the ontology repository.
- Semantic Web Portal Project
  – A portal to introduce novices users to semantic Web technology and a platform for interaction between research communities.
- Web Service Modeling Ontology (WSMO)
  - Used for describing services and its automation process. It is based on WSMF (Web Service Modelling Framework)
- SWSE - Semantic Web Search Engine
  – To continuously explore and index the current Semantic Web.
- Web Service Modeling Language (WSML)
  – The WSML focuses on developing a formal language for Semantic Web and to provide a rule-based language for the semantic Web.
- Web Service Execution Environment (WSMX)
  – Main focus is on means to achieve dynamic interoperability of Web services.

WSMO (Web Services Modeling Ontology) is developed to encompass the different aspects of Web service Development. It aims to solve the interoperability issue by means of mediators and goals (pre and post conditions). WSMO introduces semantic description into Web services by means of F-logic statements. The complexity of F-Logic can serve as a disadvantage to users who are unfamiliar with rule languages. In METEOR-S, we have research ongoing in the area of representing constraints. Constraints are represented as boolean expression in annotated source code and converting the same to SWRL rules in WSDL-S documents. The former representation enables the developers to easily understand the constraints, while the later is used for logical querying using inference engines.

10. Conclusion:

An in-depth understanding of the basics of semantics will help us take advantage of Semantic Web and its applications. Semantic Web Services, an application of Semantic Web, is gaining popularity in the field of application integration and e-commerce. Ontologies are the source of semantic

knowledge in Web Services. They help to bring about integration by providing unambiguous definition of the service, its operation and their parameters. Finally, an overview of frameworks supporting Semantic Web Services- OWL-S and DERI is presented. OWL-S is reviewed for the types of semantics used and semantic representation adopted. Section 5 discusses how the semantic information resident in a system, demonstrates the capability of a system to extend support for automation. An overview of research groups in DERI to enable Semantic Web Services is also presented. The essay outlines some salient research issues like round-tripping in type-systems, typing of semantic types and validation of annotations, areas which needs to be addressed for the success of Semantic Web and related applications.

11. Appendix

Future work in automated annotation may wish employ the use functional languages such as Haskell to suggest annotations. For example, based on the classes/concepts used to annotate the input and output, the functional concept to represent the operations can be inferred. Haskell uses a type inferencing mechanism called the *Hindley-Milner* type system [Arvind,2002]. An example for type inference is as follows [Verity,2002]

```
sumlist [] = 0  ------- [1]
sumlist (h:t) = h + sumlist t  --------[2]
```

The inferred type of the elements in 'sumlist' ([] indicates argument pattern) by Haskell is,

```
sumlist:: [Int] -> Int
```

The type inference performed by Haskell is explained as follows; Statement [1] has return type '0', which infers the return type of function 'sumlist' to be 'Int'. The element head 'h' is added to the result of 'sumlist' using the pattern (h:t) ,which means that the element type of the list should be 'Int' to give the appropriate return type of 'Int'. If the function elements were of type other than 'Int' the compiler would throw errors, as the inferred type of 'sumlist' elements and the input type of 'sumlist' elements vary. All type checking is performed during compilation, and this reduces errors due to type-mismatch during execution time. Type transformations will help in choosing the closest possible semantic definition and extending it to address application specific needs. Type inferencing similar to Haskell can be used to suggest annotations while developing Semantic Web services.

12. Recommended Reading

[1] Where are the Semantics in the Semantic Web? By Michael Uschold. http://www.starlab.vub.ac.be/WhereAreSemantics-AI-Mag-FinalSubmittedVersion2.pdf

[2] Introduction to Formal Semantics By Gerhard Jaager. http://www.phil-fak.uni-duesseldorf.de/summerschool2002/Jaeger1.pdf

[4] Model Theory - http://plato.stanford.edu/entries/model-theory/

13. References

[Aggarwal et al, 2004] R. Aggarwal, K. Verma, A. Sheth, J.Miller and W. Milnor, Constraint Driven Web Service Composition in METEOR-S, Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004), Shanghai, China (September 2004) pp. 23-32

[Arvind, 2002] T. Arvind , Lecture talk on The Hindley-Milner Type System, Laboratory for Computer Science, MIT, http://ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-827Multithreaded-Parallelism-Languages-and-CompilersFall2002/EC7C6358-B09C-4BF7-9BA1-6C6F7B4EF5E9/0/L06HindleyMilnerPrint.pdf

[Berners-Lee et al., 2001] T. Berners-Lee, J. Handler, and O. Lassila: The Semantic Web, Scientific American, 284(5):34--43, May 2001.

[Boley et al., 2004] H. Boley, M. Dean, B. Grosof, M. Sintek, B. Spencer, S.Tabet and G. Wagner., The Rule Markup Initiative, http://www.ruleml.org, 2004

[Browne, 2002] C. Browne, Functional Programming Languages, http://cbbrowne.com/info/functional.html

[Chinnici et al., 2004] R. Chinnici et al., Web Services Description Language (WSDL) 2.0, available at http://www.w3.org/TR/2004/WD-wsdl20-20040326/, 2004

[Chinnici et al., 2003] R. Chinnici et al., Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language

[DERI, 2003] DERI International, http://www.deri.org/research/groups/

[Fensel et. al., 2002] D. Fensel, C. Bussler, Y. Ding, V. Kartseva, M. Klein, M. Korotkiy, B. Omelayenko, and R. Siebes: Semantic Web Application Areas. In Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems, Stockholm - Sweden, June 27-28, 2002

[Fensel & Musen, 2001] D. Fensel and M. Musen, Special Issue on Semantic Web Technology, IEEE Intelligent Systems (IEEE IS), 16(2), 2001.

[Ghallab,1998] M. Ghallab et al., PDDL-The Planning Domain De Language V. 2. Technical Report, report 1999. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[Gunter, 1992] T. Gunter, Semantics of Programming Languages: Programming Techniques. The MIT Press, Cambridge, Massachusetts.

[Haarslev & Möller, 2001] R. Haarslev and V. Möller, RACER- Description of the RACER System and its Applications Proceedings of International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August 2001

[Hawke, 2002]S. Hawke, The Semantic Web -http://www.w3.org/2002/03/semweb/

[Horrocks et al., 2004] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, Draft Version 0.6 of 23 March 2004 http://www.daml.org/rules/proposal

[Horrocks et. al., 2003] I. Horrocks, P. Schneider,F. Harmelen, From SHIQ and RDF to OWL: The Making of a Web Ontology Language of Web Semantics, 1(1):7-26, 2003.

[Eijck, 2003]J. Eijck, Computational Semantics and Type Theory, Chapters 1-6, http://homepages.cwi.nl/~jve/cs/cs.pdf

[Farrugia, 2003]J. Farrugia, Model-Theoretic Semantics for the Web, Proceedings of WWW2003, May 20-24, 2003, Budapest, Hungary.ACM 1-58113-680-3/03/0005.

[Gruber, 1993] T.Gruber, A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220, 1993

[JDK1.5, 2004] jdk 1.5 Java Development Kit-http://java.sun.com/j2se/1.5.0/index.jsp

[Jurafsky & Martin, 2000] D. Jurafsky and J. Martin, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice Hall Publishing, 2000

[Kalyanpur et al., 2004] A. Kalyanpur, D. Pastor, S. Battle and J. Padget, Automatic mapping of OWL ontologies into Java http://www.mindswap.org/aditkal/www2004_OWL2Java.pdf.

[Klyne & Carroll, 1998] G. Klyne and J. Carroll, KIF- Knowledge Interchange Format: Draft proposed American National Standard (dpans). Technical s Report 2/98-004, ANS, 1998.

[Lagoze, 2002] C. Lagoze, Lecture notes on 'Semantic Web Ontologies and Data Models', http://www.cs.cornell.edu/Courses/cs502/2002SP/lectures/lecture%203-07-02.ppt

[McGuinness & Harmelen, 2004] D. McGuinness and F. Harmelen, OWL Web Ontology Language Overview- http: //www.w3.org/ TR/2004/RECowl- features-20040210/

[METEOR-S, 2002] METEOR-S: Semantic Web Services and Processes, http://swp.semanticweb.org, (2002).

[Mihic & Trezzo, 2002] M. Mihic and J. Trezzo, JSR 181 Java Specification Requests- http://www.jcp.org/en/jsr/detail?id=181.

[Nivelle, 2000] H. Nivelle, Lecture Notes on 'Resolution based Theorem Proving', http://www.mpi-sb.mpg.de/~nivelle/talks/wroclaw2000_algemeen.pdf

[OWL-S, 2003] OWL-S Specification –http://www.daml.org/services/owl-s/1.1

[OWL-S, 2004] OWL-Services, http://www.w3.org/2004/Talks/0430-bp-owl/OWL-S.pdf

[OWL-S, 2004a] OWL-S : Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.1B/owl-s.ps

[Paolucci et al., 2002a] M.Paolucci, T.Kawamura, T.Payne and K. Sycara, Semantic Matching of Web Services Capabilities, " Proceedings of the ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002. Springer 2002.

[Rajasekaran et al., 2004] P. Rajasekaran, J.Miller, K. Verma and A.Sheth, Enhancing Web Services Description and Discovery to Facilitate Composition, Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04), Part of the 2nd International Conference on Web Services (ICWS'04), San Diego, California (July 2004) pp. 34-47.

[Sivashanmugam et al., 2003] K. Sivashanmugam, K., Verma, A. Sheth and J. Miller, Adding Semantics to Web Services Standards, Proceedings of 1st International Conference of Web Services, 395-401, (2003).

[WSDL-S, 2004] WSDL-S files: http://lsdis.cs.uga.edu/~projects/meteor-s/wsdl-s