Advances in Internet of Things, 2015, 5, 15-20
Published Online April 2015 in SciRes. http://www.scirp.org/journal/ait
http://dx.doi.org/10.4236/ait.2015.52003

Web Semantic and Ontology

Elodie Marie Gontier

Professor of French and History, Paris, France
Email: cgontier06@aol.com

Received 18 March 2015; accepted 26 June 2015; published 29 June 2015

Abstract

Ontologies have become a popular research topic in many communities. In fact, ontology is a main component of this research; therefore, the definition, structure and the main operations and applications of ontology are provided. Web content consists mainly of distributed hypertext and hypermedia, and is accessed via a combination of keyword based search and link navigation. Hence, the ontology can provide a common vocabulary, and a grammar for publishing data, and can supply a semantic description of data which can be used to preserve the ontologies and keep them ready for inference. This paper provides basic concepts of semantic web, and defines the structure and the main applications of ontology.

Keywords

Ontology, Semantic Web, Language OWL

1. What Do We Represent in an Ontology?

In the context of Semantic Web, ontologies describe domain theories for the explicit representation of the semantics of the data. In other words, ontology should be seen as a right answer to provide a formal conceptualization. Indeed, ontology must translate an explicit consensus and develop a certain level of division. It has two essential aspects to allow the operation of the resources of web by various applications or software agents. The ontologies serve then:

1) For the vocabulary, the structuring and the operation of metadatas;

2) As representation pivot for the integration of springs of heterogeneous data;

3) To describe the web departments, and generally, everywhere it is going to be necessary to press software modules on semantic representations requiring certain consensus.

Ontology (def. 1): all the objects recognized as existing in the domain. To build an ontology, it is also to decide on the way of being and to exist objects. To continue towards a definition of ontology, it seems to us essential to remind that the works on the ontologies are developed in an IT context that is the case, for

Engineering of knowledge, Artificial intelligence or, more specifically here, the context of Semantics Web where the final goal is to specify an IT artefact. In this context, the ontology becomes then a model of the existing objects which makes a reference to it through concepts of the domain.

The developments are a free performance of the reasons adduced for the works of Guarino and Giaretta [1]. They aim at progressing towards a definition reporting an evolutionary process of construction.

Ontology (def. 2): an ontology involves or includes a certain worldview compared with a given domain. This sight is often conceived as a set of concept—e.g. entities, attributes, and process—their definitions and their interrelations. We call it a "conceptualization". An ontology can take various forms, but it will include inevitably a vocabulary of terms and specification of their meaning. So, it is a specification partially reporting a conceptualization. This second definition proposes another point of view compared with the first one, coherent with her but more precise, in terms of specification and compared with web operation.

Ontology is good at conceptualization, like said Thomas Gruber "it's an explicit specification of conceptualization":

1) Afterward,it must be used in an IT artefact, but we have to specify it more later. Ontology will also have to be a logical theory for which we shall specify the manipulated vocabulary;

2) Finally, the conceptualization is sometimes specified in a very precise way. That's why a logical theory cannot always report it in an exact way: she can accept the interpretative wealth of the domain conceptualized in an ontology and make it thus only partially. This gap between the conceptualization and the formal specification is described by Guarino as the ontological commitment which the designer has to accept in the passage of the one to the other one.

The ontology is a theory on the representation of the knowledge. As indicated in 2000, the ontology "defines the kinds of things that exist in the application domain". It is by this theory that it unifies in the domain of the computing. The ontology "is a formal, explicit specification of a shared conceptualization" (Gruber, 1993). For the IT specialist of semantic web, the ontology is a consensual model, because the conceptualization is shared and brings then to build a linguistic specification with the vocabulary RDF/RDFS and the language OWL. In the semiotic perspective, conceptualization according to Gruber relates to the domain of the speech because it is the abstraction. The domain of the speech takes place of the referent. In semiotics, we shall say that the ontology symbolizes the conceptualization, the terms, the notions and the relations which are conceptualized.

2. The Web Ontology Language Owl

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 [2]. Its URI-based vocabulary and XML-based grammar are key enablers to web scale knowledge management and sharing.

One of the strong results of semantic web on the ontologies is the normalization of their expression. This point, essential if we want that the ontologies can be shared, exactly seems to find a solution in the context of semantic web: the definition of the language OWL (Web Ontologies Language) at various levels of complexity (capacity of complexity of the descriptions versus calculability) is the best example.

Although already recognisable as an ontology language, the capabilities of RDF are rather limited: they do not, for example, include the ability to describe cardinality constraints (such as Hogwarts Students having at most one pet), a feature found in most conceptual modelling languages, or to describe even a simple conjunction of classes.

The need for a more expressive ontology language was widely recognised within the nascent semantic web research community, and resulted in several proposals for "web ontology languages", including SHOE, OIL and DAML + OIL. The architecture of the web depends on agreed standards and, recognising that an ontology language standard would be a prerequisite for the development of the semantic web, the World Wide Web Consortium (W3C) set up a standardisation working group to develop a standard for a web ontology language. The result of this activity was the OWL ontology language standard [3]. OWL exploited the earlier work on OIL and DAML + OIL, and also tightened the integration of these languages with RDF. The integration of OWL with RDF includes the provision of a RDF based syntax. This has the advantage of making OWL ontologies directly accessible to web based applications, but the syntax is rather verbose and not easy to read. For example, the description of the above mentioned class of Student Wizards would be written in RDF/XML as:

```
<owl:Class>
<owl:intersectionOf
rdf:parseType="Collection">
<owl:Class rdf:about="#Student"/>
<owl:Class rdf:about="#Wizard"/>
</owl:intersectionOf>
</owl:Class>
```

In the remainder of this paper, I will instead use an informal \human readable" syntax based on the one used in the Protege 4 ontology development tool [4]. A key feature of OWL is its basis in Description Logics (DLs), a family of logic-based knowledge representation formalisms that are descendants of Semantic Networks and KLONE, but that have a formal semantics based on rstorder logic [5]. These formalisms all adopt an objecto-riented model, similar to the one used by Plato and Aristotle, in which the domain is described in terms of individuals, concepts (called classes in RDF), and roles (called properties in RDF). Individuals, e.g., "Hedwig", are the basic elements of the domain; concepts, e.g., "Owl", describe sets of individuals having similar characteristics; and roles, e.g., "hasPet", describe relationships between pairs of individuals, such as "HarryPotter hasPet Hedwig".

In order to avoid confusion, I will keep to the already introduced RDF terminology and from now on refer to these basic language components as individuals, classes and properties. As well as atomic class names such as Wizard and Owl, DLs also allow for class descriptions to be composed from atomic classes and properties. A given DL is characterised by the set of constructors provided for building class descriptions. OWL is based on a very expressive DL called SHOIN (D) a sort of acronym derived from the various features of the language [6]. The class constructors available in OWL include the Booleans and, or and not, which in OWL are called intersectionOf, unionOf and complement Of, as well as restricted forms of existential and universal quantication, which in OWL are called, respectively, "some Values From" and "all Values From" restrictions. OWL also allows for properties to be declared to be transitive| if has Ancestor is a transitive property, then Enoch has Ancestor Cain and Cain has Ancestor Eve implies that Enoch has Ancestor Eve. The S in SHOIN (D) stands for this basic set of features.

In OWL, some values from restrictions are used to describe classes whose instances are related, via a given property, to instances of some other class. For example, Wizard and hasPet some Owl describes those Wizards having pet Owls. Note that such a description is itself a class, the instances of which are just those individuals that satisfy the description; in this case, those individuals that are instances of Wizard and that are related via the hasPet property to an individual that is an instance of Owl. If an individual is asserted to be a member of this class, then we know that they must have a pet Owl, although we may not be able to identify the Owl in question, *i.e.*, some values from restrictions specify the existence of a relationship. In contrast, all values from restrictions constrain the possible objects of a given property and are typically used as a kind of localised range restriction.

For example, we might want to state that Hogwarts students can have only Owls, Cats or Toads as pets without placing a global range restriction on the hasPet property (because other kinds of pet may be possible in general). We can do this in OWL as follows:

Wizard and hasPet some Owl

describes those Wizards having pet Owls. Note that such a description is itself a class, the instances of which are just those individuals that satisfy the description; in this case, those individuals that are instances of Wizard and that are related via the hasPet property to an individual that is an instance of Owl. If an individual is asserted to be a member of this class, then we know that they must have a pet Owl, although we may not be able to identify the Owl in question, *i.e.*, some values from restrictions specify the existence of a relationship. In contrast, all values from restrictions constrain the possible objects of a given property and are typically used as a kind of localised range restriction. For example, we might want to state that Hogwarts students can have only Owls, Cats or Toads as pets without placing a global range restriction on the has Pet property (because other kinds of pet may be possible in general). We can do this in OWL as follows:

Class: HogwartsStudent

SubClassOf: hasPet only (Owl or Cat or Toad)

In addition to the above mentioned features, OWL also allows for property hierarchies (the H in SHOIN (D)), extensionally denied classes using the one of constructor (O), inverse properties using the inverse of property constructor (I), cardinality restrictions using the minCardinality, maxCardinality and cardinality constructors (N), and the use of XML Schema datatypes and values (D) [7]. For example, we could additionally state that the instances of Hogwarts House are exactly Gryndor, Slytherin, Ravenclaw and Huepu, that Hogwarts students have an email address (which is a string) and at most one pet, that isPetOf is the inverse of hasPet and that a Phoenix can only be the pet of a Wizard:

Class: HogwartsHouse
EquivalentTo: {Gryffindor, Slytherin
Ravenclaw, Hufflepuff}
Class: HogwartsStudent
SubClassOf: hasEmail some string
SubClassOf: hasPet max 1
ObjectProperty: hasPet
Inverses: isPetOf
Class: Phoenix
SubClassOf: isPetOf only Wizard

An OWL ontology consists of a set of axioms. As in RDF, subClassOf and subPropertyOf axioms can be used to dene a hierarchy of classes and properties. In OWL, an equivalent Class axiom can also be used as an abbreviation for a symmetrical pair of subClassOf axioms. An equivalentClass axiom can be thought of as an "if and only if" condition: given the axiom C equivalentClass D, then an individual is an instance of C if and only if it is an instance of D. Combining subClassOf and equivalentClass axioms with class descriptions allows for easy extension of the vocabulary by introducing new names as abbreviations for descriptions. For example, the following axiom:

Class: HogwartsStudent
EquivalentTo: Student and attendsSchool
value Hogwarts

introduces the class name HogwartsStudent, and asserts that its instances are just those Students that attend Hogwarts. Axioms can also be used to state that a set of classes is disjoint, and to describe additional characteristics of properties: as well as being Transitive, a property can be Symmetric, Functional or Inverse Functional. For example, the axioms:

DisjointClasses: Owl Cat Toad
Property: isPetOf
Characteristics: Functional

state that Owl, Cat and Toad are disjoint (*i.e.*, that they have no instances in common), and that isPetOf is Functional (*i.e.*, pets can have at most one owner). The above mentioned axioms describe constraints on the structure of the domain, and play a similar role to the conceptual schema in a database setting; in DLs such a set of axioms is called a TBox (Terminology Box). OWL also allows for axioms asserting facts about some concrete situation, similar to data in a database setting; in DLs such a set of axioms is called an ABox (Assertion Box). These might, for example, include the facts:

Individual: HarryPotter
Types: HogwartsStudent
Individual: Fawkes
Types: Phoenix
Facts: isPetOf Dumbledore

Basic facts (*i.e.*, those using only atomic classes) correspond directly to RDF triples|the above facts, for example, correspond to the following triples:

HarryPotter rdf:type, HogwartsStudent
Fawkes rdf:type Phoenix
Fawkes isPetOf Dumbledore

The term ontology is often used to refer just to a conceptual schema or TBox, but in OWL an ontology can consist of a mixture of both TBox and ABox axioms; in DLs, this combination is known as a Knowledge Base. Description Logics are fully edged logics and so have a formal semantics. DLs can, in fact, be seen as decidable subsets of rst-order logic, with individuals being equivalent to constants, concepts to unary predicates and roles to binary predicates. As well as giving a precise and unambiguous meaning to descriptions of the domain, this also allows for the development of reasoning algorithms that can provide correct answers to arbitrarily complex queries about the domain. An important aspect of DL research has been the design of such algorithms, and their implementation in (highly optimised) reasoning systems that can be used by applications to help them "understand" the knowledge captured in a DL based ontology.

3. Ontology Language Processors

As we can see, ontologies are like taxonomies but with more semantic relationships between concepts and attributes; they also contain strict rules used to represent concepts and relationships. An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base. According to this definition, the same ontology can be used for building several knowledge bases.

Indeed, an ontology construct conveys descriptive semantics, and its actionable semantics is enforced by inference. Hence, effective tools, such as parsers, validators, and inference engines, are needed to fulfill the inferenceablity objective:

1. **OWLJessKB** is the descendent of DAMLJessKB and is based on the Jess Rete inference engine [7].
2. **Java Theorem Prover (JTP)** developed at Stanford university [8] supports both forward and backward chaining inference using RDF/RDFS and OWL semantics.
3. **Jena** (http://jena.sourceforge.net/), developed at HP Labs at Bristol, 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 [9]
4. **F-OWL** developed at UMBC, is an inference engine which is based on Flora-218 [10].
5. **FaCT ++** uses the established FaCT algorithms, but with a different internal architecture. Additionally, FaCT ++ is implementated using C ++ in order to create a more efficient software tool, and to maximise portability [11].
6. **Racer** (https://www.ifis.uni-luebeck.de/index.php?id=385) is a description logic based reasoner. It supports inference over RDFS/DAML/OWL ontologies through rules explicitly specified by the user [12].
7. **Pellet** (http://www.w3.org/2004/04/13-swdd/SwoopDevDay04.pdf), developed at the University of Maryland, is a "hybrid" DL reasoner that can deal both TBox reasoning as well as non-empty ABox reasoning [13]. It is used as the underlying OWL reasoner for SWOOP ontology editor [14] and provides in-depth ontology consistency analysis.
8. **TRIPLE** developed by Sintek and Decker into *Proceedings of the* 1*st International Semantic Web Conference* [15], 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.
9. **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 a various other rule languages and ontologies (implicit axioms).

The semantics conveyed by ontologies can be as simple as a database schema or as complex as the background knowledge in a knowledge base. By using ontologies in the semantic web, users can leverage the advantages of the following two features:

1) Data are published using common vocabulary and grammar;

2) The semantic description of data is preserved in ontologies and ready for inference.

Ontology transformation [16] is the process used to develop a new ontology to cope with new requirements made by an existing one for a new purpose, by using a transformation function t. In this operation, many changes are possible, including changes in the semantics of the ontology and changes in the representation formalism. Ontology Translation is the function of translating the representation formalism of an ontology while keeping the same semantic. In other words, it is the process of change or modification of the structure of an ontology in order to make it suitable for purposes other than the original one.

There are two types of translation. The first is translation from one formal language to another, for example from RDFS to OWL, called syntactic translation. The second is translation of vocabularies, called semantic translation [17]. The translation problem arises when two Web-based agents attempt to exchange information, describing it using different ontologies. The goal of an ontology is to achieve a common and shared knowledge that can be transmitted between people and between application systems. Thus, ontologies play an important role in achieving interoperability across organizations and on the semantic web, because they aim to capture domain knowledge and their role is to create semantics explicitly in a generic way, providing the basis for agreement within a domain. Thus, ontologies have become a popular research topic in many communities. In fact, ontology is a main component of this research; therefore, the definition, structure and the main operations and applications of ontology are provided.

4. Conclusion

Ontologies play an important role in achieving interoperability across organizations and on the semantic web, because they aim to capture domain knowledge and their role is to create semantics explicitly in a generic way, providing the basis for agreement within a domain. In other words, the current web is transformed from being machine-readable to machine-understandable. So, ontology is a key technique with which to annotate semantics and provide a common, comprehensible foundation for resources on the semantic web.

References

[1] Guarino, N. and Giaretta, P. (1995) Ontologies and Knowledge Bases. In: *Towards Very Large Knowledge Bases*, IOS Press, Amsterdam, 1-2.

[2] Web Ontology Language (OWL) Offers Additional Knowledge Base Oriented Ontology Constructs and Axioms. http://www.w3.org/2002/Talks/04-sweb/slide12-0.html

[3] Ian Horrocks, Ontologies and the Semantic Web, Oxford University Computing Laboratory.

[4] http://protege.stanford.edu/

[5] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P.F., Eds. (2003) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge.

[6] Horrocks, I. and Sattler, U. (2007) A Tableau Decision Procedure for SHOIQ. *Journal of Automated Reasoning*, **39**, 249-276.

[7] Joseph, K. and William, R. (2003) DAMLJessKB: A Tool for Reasoning with the Semantic Web. *IEEE Intelligent Systems*, **18**, 74-77.

[8] Joseph, K. And William, R. (2003) DAMLJessKB: A Tool for Reasoning with the Semantic Web. *IEEE Intelligent Systems*, **18**, 74-77.

[9] Richard, F., Jessica, J. and Gleb, F. (2003) JTP: A System Architecture and Component Library for Hybrid Reasoning. Stanford University, Stanford.

[10] Carroll, J.J, Ian, D., Chris, D., Dave, R., Andy, S. and Kevin, W. (2004) Jena: Implementing the Semantic Web Recommendations. *Proceedings of the* 13*th International World Wide Web Conference on Alternate Track Papers & Posters*, 2004, 74-83. ISBN:1-58113-912-8.

[11] Zou, Y.Y., Finin, T. and Chen, H. (2004) F-OWL: An Inference Engine for the Semantic Web. Formal Approaches to Agent-Based Systems. Vol. 3228 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. *Proceedings of the Third International Workshop* (*FAABS*), 16-18 April 2004.

[12] Dmitry, T. and Ian, H. (2003) Implementing New Reasoner with Datatypes Support. Wonder Web: Ontology Infrastructure for the Semantic Web Deliverable.

[13] Ian, H. (1998) The FaCT System. Automated Reasoning with Analytic Tableaux and Related Methods. *International Conference Tableaux*-98, Springer Verlag, Berlin, 307-312.

[14] Evren, S. and Bijan, P. (2004) Pellet: An OWL DL Reasoner. In: *Description Logics*, CEUR-WS.org, 9.

[15] Aditya, K., Bijan, P. and James, H. (2005) A Tool for Working with Web Ontologies. *International Journal on Semantic Web and Information Systems*, **1**, 4.

[16] Michael, S. and Stefan, D. (2002) TRIPLE—A Query, Inference, and Transformation Language for the Semantic Web. *Proceedings of the* 1*st International Semantic Web Conference* (*ISWC*-02), Springer-Verlag, Berlin, 364-378.

[17] Chalupsky, H. (2000) OntoMorph: A Translation System for Symbolic Knowledge. *Proceedings of KR*, Morgan Kaufmann Publishers, San Francisco, 471-482.