Chapter

Ontology Language XOL Used for Cross-Application Communication

Jinta Weng, Jing Qiu and Ying Gao

Abstract

The 2000s may be the flourishing time of the topic of ontology. Specialists and scholars concentrated to define ontology effectively and formulated uniform ontology protocol. Ontology language can be classified into SHOE, OML, XOL, OIL, OWL, and RDFs by different protocols and syntaxes. As for effective exchange of the different ontology messages in different applications, US bioinformatic community and researcher develop a XML-based ontology language. With the simplified OKBC-Lite protocol and flexible XML syntax, XOL offers the ways to define an ontology with the human-readable XML, simplified protocol, and compatible interface. In this chapter, we will introduce its motivation from history, orientation in development, semantic usage, and interpreted example in detail.

Keywords: ontology exchange language, Ontolingua, XOL, open knowledge base, Semantic Web

1. Introduction

Internet had maintained a rapid development between the 1990s and 2000s, which not only gives birth to various applications, abundant network facilities, and diverse websites but also accelerates the next generation of Semantic Web. After Berners-Lee put forward the imaginary structure of Semantic Web in 1998, W3C with many semantic work teams is dedicated to develop the technical standard of Resource Description Framework [1]. As ontology is the essence and basic of a resource, technical combinations of paradigm and languages are used to define it.

1.1 Background

Knowledge engineering has become an essential part of expert system in artificial intelligence. It is important to define the specific knowledge, also known as domain database or knowledge base, for multiple applications. However, traditional knowledge base just reveals the key and value of the data, thus paying less attention on ontology.

Ontology is the description and formulization of thing. By full-semantic and expressive ontology, more information and relationship are able to excavate. In order to build more humanistic and intelligent system, scholars had developed different ontology languages. Although many ontology languages give methods to solve the ontology definition. However, a new language or ontology protocol should also be formulated to deal with the cross-application problem.

1.2 Motivation

Accompanying with the development of Internet, more infrastructure, application, and knowledge base are generated. In normal knowledge supported systems, domain expert will first considerate the software environment and self-knowledge background and then choose the suitable knowledge scheme and ontology for the system. However, when it comes to the cross applications or large knowledge-assisted system, ontologies in system need to be reused. First, knowledge scheme in different systems may exist difference from the expert's personal cognize. Second, it offers several ontology languages for each system; thus, different ontology schemes can show in different formats, which make it hard to communicate in different applications. Third, the increasing demand of openness and the sharing lead of ontology could be exchange. Therefore, an ontology exchangeable protocol or new ontology language supporting to exchange should be redefined.

To realize the need of an evaluation on ontology in bioinformatics, several researchers on the US evaluation team developed a new specific ontology language—XOL [2]. By flexible XML expression and simplified protocol, XOL (xml-based ontology exchange language) is able to express and exchange different ontology information across incompatible applications.

1.3 Definition

XOL is an ontology language developing for exchange ontology in cross applications. It takes inspiration from OKBC (a protocol used for open knowledge base, see in Ref. [3]) and Ontolingua (another ontology used for reusing and editing ontology, see in Ref. [4]). Its syntax is based on human-readable and high compatible XML document. XOL can also respect as one effective intermedia language in ontologies' use, exchange, negotiation, and cocreation.

1.4 A simple example

Note the following XOL definitions:

```
<class>
<name> [class-name] </name>
</class>
<slot>
<[slot-attribute]></[slot-attribute]>
</slot>
<individual>
<name></name>
<type></type>
<...></...>
</individual>
```

All of above XOL elements are pertained to all ontologies. Between the pair of <class></class> defines the basic information of this ontology, like the name of the class during the tag pair of <name></name>.

Pair of <slot></slot> will depict the attribute and restriction of the class, like value's type of the attribute and the data restriction.

The last tag <individual> </individual> will give an instance of self-class or multiclass. It is not allowed to use the subclass as the individual element.

With the human-readable and self-defined XML syntax, XOL can express the ontology in a concise way. However, it may also lead to the ontology inconformity

while using XML syntax merely or personally. A more restrictive and stationary tag OKBC-Lite was chosen soon.

2. Why is XOL based on XML?

Generally speaking, each ontology language makes up for using syntax and language protocol. To realize the essence of XOL, we will show the different classifications of ontology languages based on the syntax and semantic rules.

According to the use of syntax, we can classify the ontology languages into three types as follows:

HTML format: Hypertext Markup Language (HTML) is the basic document mark of the current web. To extend the semantic character of the HTML, ontology language like SHOE offers an effective way to support semantic annotation by more extended webpage label.

XML format: Extensive Markup Language (XML) is a more human-readable and concise document for storing and defining different data. Ontology document made by XML format can easily locate by its hierarchical structure and semantic DTD tag.

RDF format: Resource Description Framework (RDF) is a new way to define ontology after XOL. It is a resource model always accompanied by a specific URI and extended specific XML-like label to depict the relation and knowledge model between the resources. It not only specifically and strictly expresses the data but also makes the alternation, merging, and inference possible.

According to language protocol of these languages, ontology language can divide into first-order predicate logic language, frame-based language, and concept-role restriction language.

First-order predicate logic language is the most accurate and original language in knowledge representation. The predicate formula is the formula formed by joining some predicates together with the predicate join symbol, like the largest formalized language Cyclo [5] and KIF [6].

Frame-based language is a language that includes the aforehand defining framework and simplified first-order logic language. Owing to excessive strict first-order predict logic and unreadable syntax, Ontolingua and frame logic are developed to remedy this defect.

Concept-role restriction language is an effort that most language currently adopts. This type of language offers a hierarchy way to represent the hyponymy by concept and the individual's signal. It reveals the relationship and value restrictions between different ontologies by role mark, like OML [7].

To note the difference between ontology languages in multiple syntax formats, we will give a detailed introduction for some ontology language with the technical developing route.

2.1 SHOE (HTML format)

HTML had covered with a long history before the World Wide Web (WWW) appeared and is one of document standards of Standard Generalized Markup Language (SGML). SGML offers a high standard and complicated description about the document resource. As SGML is hard to learn, use, and realize, researcher put forward the HTML in 1989 after considering the computer's ability. HTML is the mere application of SGML in the WWW times. After few years of great development, HTML is widely known in the web document district. Semantic Web, as the next generation of WWW, is also the use of the HTML syntax in the ontology language. We called this simple HTML ontology language as SHOE.

SHOE is a specification that describes an extension to HTML, which provides a way to semantically describe important information about HTML or other web documents [1]. It offers a hierarchical classification mechanism for HTML documents and non-HTML documents or subsections of HTML documents. The intent of this specification is to make it possible for user agents, robots, and so on, to gather truly meaningful information about web pages and documents, enabling significantly better search mechanisms and knowledge gathering. Let us take the SHOE as an example; it can divide into two steps as follows:

Define an ontology

<Ontology "ontology-unique-name" version = "1.0" backward-compatible-with = "version list">

Use an existing XOL ontology

<Ontology-extends "ontology-unique-name" version = "Version" backward-compatible-with = "version list" URL = "location">

This is a simple way to define ontologies containing rules. Ontology simply means an ISA hierarchy of categories and a set of relations between these categories in this SHOE specification. Categories will also inherit relations defined in parent categories. However, this specification does not as yet define any other forms of relationships (transitive closures, inverses, negations, etc.) and use the complicated and human-unreadable Hypertext Markup Language as the basic syntax.

2.2 KIF

At the same time, first knowledge interchange format was proposed by American National Standard (dpANS). Though many ontology languages are still developed, researcher in Standard University begins to design a language for the intercommunication. Interchange of knowledge or ontology thought out disparate computer systems (different programmers, different languages, and other discrepancy in interknowledge sharing). KIF language is logically comprehensive with declarative semantics.

In addition to these essential features, KIF is designed to maximize the implementability and readability. KIF provides a declarative language for describing knowledge. As a pure specification language, KIF does not include commands for knowledge base query or manipulation.

2.3 GFP

GFP is first motivated by the hierarchic framework design of frame-based knowledge representation systems (FRSs) used at the Stanford Knowledge Systems Laboratory for accessing Cyc, KEE, and Epikit [8]. FRSs can contain all of the database systems and knowledge systems or other frame-like projects. It is complementally developed to support knowledge sharing. It specifies a new protocol, Generic Frame Protocol (GFP), for connecting knowledge bases (KBs) in FRSs. In more detail, it provides numbers of operations to formulate a general interface for all of the FRSs. Also, complementary tools were also produced to keep independent and general operation generation. GFP shows well compatibility between different languages, including Java, C (client implementation only), and Common Lisp. Thus, some format conversations of languages are also needed.

2.4 OKBC

After GFP coming out, OKBC, a new protocol called Open Knowledge Base Connectivity, has taken it up in more implicit knowledge model and knowledge

operation. It first uses some open ontology systems, such as EcoCyc, GKB Editor, and Ontolingua projects. With 2 years of development, OKBC quickly used in several ontology sharing projects.

OKBC handles the knowledge in more implicit representation formalism, which we called OKBC Knowledge Model in later years. This model not only supports an object-oriented representation of knowledge but also can found the commonly knowledge structure from different KRSs. Therefore, it can serve as an implicit knowledge interlingua by its powerful character in knowledge for all of the systems using OKBC.

2.5 Ontolingua

Ontolingua, which accompanies with different ontology languages breaking out, can serve as a basic framework to support open or domain knowledge sharing system. The syntax of Ontolingua definition is based on GFP. It is motivated by the need of Summer Ontology Project, a pilot study in which researchers from several groups and institutions met weekly to design ontology of terms used in modeling electromechanical devices. **Figure 1** depicts the structure of Ontolingua.

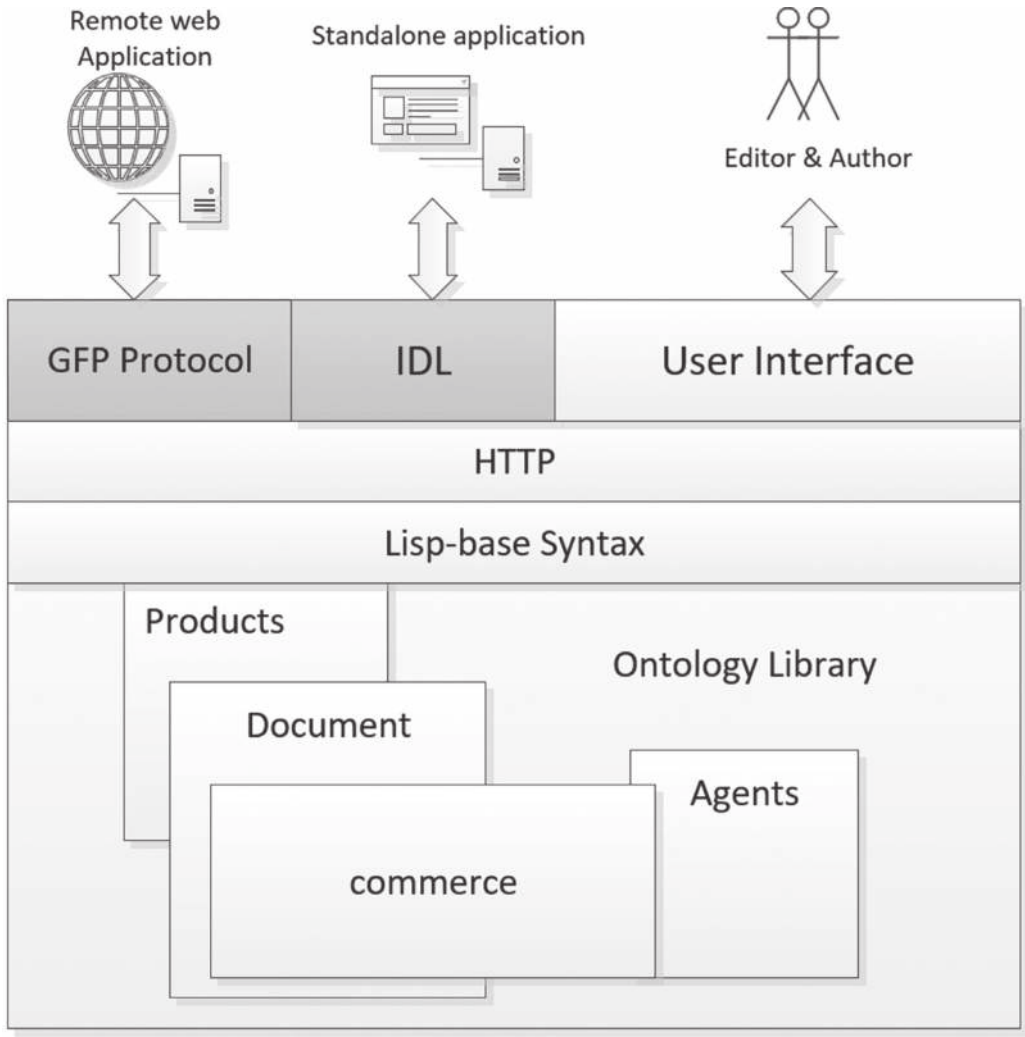

Figure 1.
The structure of Ontolingua.

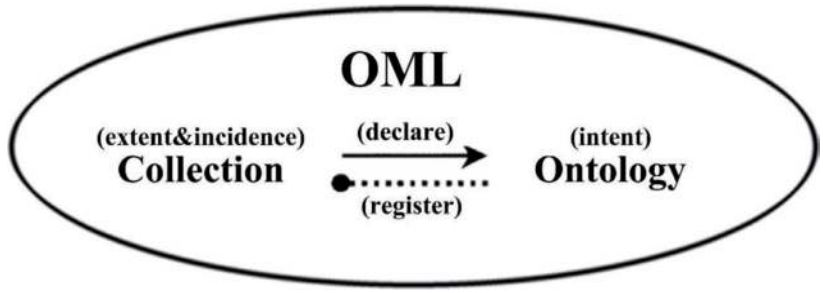

Figure 2.
The structure of OML.

At first, authors and editors will publish their ontology or maintain an ontology with a HTTP protocol connection. Users may access the different defined ontologies in Ontology Library, for example, ontologies, product ontologies, document ontologies, commerce ontologies, and agent knowledge. Beside the ontology editors or editors, some remote application may also need to duly connect this ontology library in GFP protocol to exchange or manipulate ontology. When it comes to standalone application, interface definition language with a specific and functional method will retrieve an ontology batch from ontology library. Ontolingua keeps a balance in generalization expressed GRP protocol and selectivity in some standalone application by own interface definition language (IDL).

Ontolingua can thereby be shared by multiple user and research groups using their own favorite representation systems and can be easily ported from system to system. The syntax of Ontolingua definition is simplified with some class name, argument, and documentation string.

2.6 OML

XOL is more similar to Ontolingua. However, Ontolingua using OKBC is frame-based design and less semantic expression. At the same time, a separate set of researchers is pursuing a concept-role restriction language—OML. Ontology markup language (OML) is an adaptive change language based on SHOE. Earlier versions of OML were basically an Extensive Markup Language (XML) translation of the SHOE language with suitable changes and improvements. Common elements existing in OML can be described by paraphrasing the SHOE documentation to some degree. Now OML is highly RDF Schemas compatible, although it has own solution within the namespace problem. More importantly, OML has incorporated own elements and expressiveness of conceptual graphs. As shown in **Figure 2**, by declaring and registering operation, OML can be seen as a bridging connection between Ontology and Collection, which reflects more extent and incidence of ontology.

2.7 An ontology exchange language in XML format: XOL

With the development of the relative language, protocol and definition, and the need of an Evaluation of Ontology Exchange Languages for bioinformatics, several researchers on the evaluation team are currently developing a specification of XML expression of Ontolingua using OKBC, while a separate set of researchers is pursuing a frame-based version of OML. However, Ontolingua first uses a Lisp-based syntax (rather than HTML-based or XML-based), which leads to become hard to develop and maintain, though the semantics of OKBC-Lite are extremely similar to the semantics of Ontolingua. At this background, XOL was first published in 1999.

3. The usages of XOL

The usages of XOL are based on frame-based approach. In this part, we will introduce the constituent part and its usage mode.

3.1 Basic data type

- Integer
- Floating point numbers
- Double-precision floating point numbers
- Strings
- Boolean
- Name of class

3.2 Classes

Classes are composed of entities. Entity that is not the class but an instance of a class or multiple classes is said to be Individual. Classes and Individual distinguish by whether entity is a class of another entity or not. Class entity male or class entity female is the subclass of another class entity human, a man called Joe is an Individual entity of the class entity. The following Class is the basic Class description defined in the OKBC-Lite (**Table 1**).

3.3 Slots on slots

Slot is common property of each class or instance. The attribute "Documentation" of class has an introduction to this class. When it comes to the specific KB, slot may divide into "own slot" in this KB or "template slot" inheriting from class.

Slots on slots, as shown in **Table 2**, are the several restrictions or declarations defined to this slot. Although it may be inherited from other KB or class, restriction or declaration on slots in this XOL file is exclusive.

Name	Description	Name	Description
THING	The root of the class hierarchy The superclass of every class	SYMBOL	The class of all symbols A subclass of THING
CLASS	The class of all classes	INDIVIDUAL	The class of all entities that are not classes
NUMBER	The class of all numbers A subclass of INDIVIDUAL	INTEGER	A subclass of NUMBER
STRING	The class of all text strings A subclass of INDIVIDUAL	LIST	The class of all lists A subclass of INDIVIDUAL

Table 1.
The classes of XOL [2].

Name	Function
DOMAIN	Specifies the domain of the binary relation represented by a slot frame
SLOT-VALUE-TYPE	Specifies the classes of which values of a slot must be an instance
SLOT-INVERSE	Specifies the inverse relation for a slot
SLOT-CARDINALITY	Specifies the exact number of values that may be asserted for a slot for entities in the slot's domain
SLOT-MAXIMUM-CARDINALITY	Specifies the maximum number of values that may be asserted for a slot for entities in the slot's domain
SLOT-MINIMUM-CARDINALITYNUMERIC	Specifies the minimum number of values for a slot for entities in the slot's domain
SLOT-NUMERIC-MAXIMUM	Specifies a lower bound on the values of a slot for entities in the slot's domain
SLOT-NUMERIC-MINIMUM	Specifies an upper bound on the values of a slot for entities in the slot's domain
SLOT-COLLECTION-TYPE	Specifies whether multiple values of a slot are to be treated as a set, list, or bag

Table 2.
The slots on slots of XOL.

Name	Description	Name	Description
VALUE-TYPE	Value can be class or multiclass or set of value	MINIMUM-CARDINALITY	The class of all symbols A subclass of THING
INVERSE	Describe the slot relation is reverse and value is reverse slot	NUMERIC-MINIMUM	Specifies lower bound on the number-type values of slot
CARDINALITY	Specifies the exact number of values asserted for a slot	NUMERIC-MAXIMUM	Specifies upper bound on the number-type values of a slot
MAXIMUM-CARDINALITY	Specifies the maximum number of values asserted for a slot	COLLECTION-TYPE	Specifies whether multiple values of a slot are to be treated as set/ list/bag.

Table 3.
The facet of XOL.

3.4 Acceptable facet

Different with slots on slots, facet is a restriction on the value of slot of individual. For instance, facet VALUE-TYPE and facet NUMERIC-MINIMUM describe the type of value of a slot and the minimum of value of a slot. **Table 3** is the acceptable facet defined in OKBC-Lite.

Facet can also divide into two parts:

- Own Facet (only state on current class or current KB)
- Template Facet (inherit from another class)

4. XOL example

Every XOL file must start with the following XML tab in the beginning.

<? xml version="1.0"?><!DOCTYPE module SYSTEM "module.dtd">

As the whole KB's first description, 'module section' will illuminate some information (name, type of DB, in which package, etc.) about this KB.

<Module> /*Every XOL file will start with a module mark */
<Name>name of this KB</Name>
<kb-type>which existing kb type</kb-type>
<package>self-defined package name</package>

The second section is 'class section.' In this section, we will introduce all these KB's classes or inherit from other class by the tag 'subclass-of' or 'instance-of.'

<Class>
<Name>name of class</Name>
<Documentation>String-type description</Documentation>
[Other-option-slot] (Subclass-of | instance-of | etc.)
</Class>

The third section is 'slot section,' which declares all slots in the class and slots existing slot value in an individual, such as the class-name, class-documentation, and other-option slot.

<Slot>.
<Name>name of own slot or template slot</name>
<Documentation>description</documentation>
<Domain (or other slots on slots in **Table 2**)>.</Domain>
</Slot>

The last section generally is 'individual section.' It contains all instances and their values in each slot. Also, it declares restriction of value of slot and slot-values.

<Individual>
<Name></name>
<instance-of></instance-of>
<Slot-values>
<Name>… </name>
<Values>…</values>
<value-type (or facet in **Table 3**)>…</value-type>
</slot-values>
</individual>

At last, remember that XOL file must use </model> to note the ending.

5. Future developments: OIL

Framework representation is to express the concepts, instances, classes, and relationships used in ontology in the form of framework. XOL is such a framework method-based ontology representation language. Unlike the rich expressions in logic-based approach, XOL leads to the deficiency in reasoning ability. The main differences stem from the fact that frames generally provide quite a rich set of primitives but impose very restrictive syntactic constraints on how primitives can be combined and on how they can be used to define a class.

Due to the deficiency of XOL in grammatical reasoning and the continuous development of DL notation, another new ontology interactive language OIL is defined [9]. It is not only an ontology description language but also a frame-based web language and an XML and RDF compatible ontology language. Its appearance unifies the characteristics of traditional ontology language and endows the new object into the inference layer.

6. Conclusion

The emergence of XOL was inspired by existing ontology and protocol, for example, SHOE, KIF, GFP, OKBC, and so on. XOL is a bridge language, which let the ontology using frame-based approach can be expressed in a simplifier way during the XML file. By the human-readable XML and unified Label limitation, it can use as an ontology exchange language during the cross application, which allows to obey the use of the XOL (in fact is OKBC-Lite) restriction. However, lacking of inferential capability and more logical expression, XOL was replaced by the subsequent ontology language OIL considering the multilanguage and more logic restriction that enable to validate ontology.

This chapter gives an overlook of XOL from the historical development across the different ontology languages. Note that XOL is not the first language in defining the ontology language for exchange date. It merely complements XML syntax and uses a simple frame-based OKBC protocol. However, it still lacks more compatible with multiple ontology protocols and different syntax. Without the more consideration into inference, ontology quickly would replace by stronger ontology system.

We also found that while designing a better widely used ontology language, we should keep a right balance between generality and specificity or between compatibility and limitation. We will focus more on the result comparison between different ontology methods and the humanity background within different languages.

Author details

Jinta Weng, Jing Qiu* and Ying Gao
Guangzhou University, China

*Address all correspondence to: qiujing.ch@gmail.com

References

[1] Hendler J, Berners-Lee T. From the semantic web to social machines: A research challenge for AI on the world wide web. Artificial Intelligence. 2010;**174**(2):156-161. DOI: 10.1016/j.artint.2009.11.010

[2] Karp R, Chaudhri V, Thomere J. XOL: An XML-based ontology exchange language. Version 0.4. 2002. Available from: http://www.ai.sri.com/~pkarp/xol [Accessed: 18 May 2019]

[3] Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP. OKBC: A programmatic foundation for knowledge base interoperability. In: Proceedings of the National Conference on Artificial Intelligence. 1998

[4] Gruber TR. Ontolingua: A mechanism to support portable ontologies. Knowledge Systems Laboratory. Stanford, CA: Computer Science Department, Stanford University; 1992:94305

[5] Lenat DB, Guha RV. Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project. 1st ed. USA: Addison-Wesley Longman Publishing Co.; 1989. DOI: 10.5555/575523

[6] Michael R. Genesereth. Knowledge interchange format [Internet]. 1998. Available from: http://logic.stanford.edu/kif/dpans.html [Accessed: 18 May 2019]

[7] Kent RE. Conceptual knowledge markup language: An introduction. NETNOMICS. 2000;**2**(2):139-169. DOI: 10.1023/a:1019186729572

[8] Vinay K. Chaudhri, et al. Generic Frame Protocol (GFP). 1997. Available from: http://www.ai.sri.com/~gfp/ [Accessed: 18 May 2019]

[9] van Harmelen F, Horrocks I. FAQs on OIL: The ontology inference layer. IEEE Intelligent Systems. 2000;**15**(6):69-72