Evolution of Ontologies and Types

Thierry Despeyroux

EVOLUTION OF ONTOLOGIES AND TYPES

Thierry Despeyroux
Inria – Paris-Rocquencourt
Domaine de Voluceau
B.P. 105
F-78153 Le Chesnay Cedex - France

ABSTRACT

Ontologies are heavily used in the context of the Semantic Web (Berners-Lee 1998, 2001) to formalize human knowledge. Ontologies engineering is now an important activity, and specialized softwares are developed to help in managing huge ontologies. The development of ontologies and of information systems can be compared to the development of programs. In this paper we make a parallel between ontologies and types in programming languages, and we use a small example to show that an ontology can be seen as a type system. When an ontology evolves, studying the impact of this evolution on the semantic annotations that use this ontology can be viewed as a type-checking process. The next step should be to import some notions used in the types community as overloading, polymorphism, type parameters, etc. to improve or create more powerful ontology definition languages.

KEYWORDS

Ontology, Type, Information system, Semantic-Web.

1. INTRODUCTION

Ontologies become more and more necessary and useful to formalize human knowledge, the size of these ontologies (in particular definitions), leads to the emergence of ontology engineering to allow the management of the creation, modification and development of huge ontologies. Some specialized software as editors are helpful (Protégé for example), and sometime database technology are required (Weithöner and al. 2004). In many aspects, the development of formal systems as ontologies is closed to the development of programs as it is already the case for web-based information systems (Pressman et al. 1998, Murugesan et al. 2001, Despeyroux 2004).

Traditional development of ontologies description formalisms refer to the world of logic (Fensel et al. 1998, Horrocks et al. 2003, Grosof et al. 2004, Patel-Schneider 2006).

In this paper we show that ontologies can be seen as type systems. Type systems have been heavily studied in the context of the semantics of programming languages (Gunter 1992). Thus it should be possible to reuse many results coming from the types community to create more powerful and more secure definition formalisms to describe human knowledge, and this is of course not incompatible with the logical vision.

Following (Luong et al. 2006, 2007) we take a small ontology and use it by means of semantics annotations, then modifying this ontology we see the effects of this evolution of the ontology on the semantics annotations. In a second step we endorse the suit of a programmer and will "implement" the same ontology as a type system, applying then the same evolution.

2. AN ONTOLOGY AND ITS EVOLUTION

An ontology is a way of representing knowledge in a formal manner. One main goal of an ontology is to be shared between a group of people to fix a terminology and the relation between concepts. Ontologies are heavily used in semantic annotations to ease both human and machine interface and is one of the foundation of the Semantic Web.

An important word in this definition is “shared”. It implies the fact that the set of definitions in an ontology is a reference (Gruber T. R., 1993). An ontology in the context of an information system can be compared to declarations or libraries in a programming context.

As for web pages, the evolution of ontologies and of semantic annotations must be controlled. The process of evolution is studied for example in (Luong 2007, Luong et al. 2007, Luong and Dieng-Kuntz 2007).

Let's take as example the ontology found in (Luong et al. 2007) described using RDF(S) (Klyne and Carroll 2004).

```
Concepts: Person, Trainee, PhdStudent, Manager, Researcher, Director, Team,
    Project;
Researcher is-a Manager;
Director is-a Manager;
Manager is-a Person;
PhdStudent is-a Person:
Trainee is-a Person;

Properties: work, manage;
Person work Team;
Manager manage Project;
```

Let’s take a list of semantic annotations, given as RDF triplets:

```
1. (r1 work v1)(r1 type Person)
2. (r2 work v2)(r2 type PhdStudent)
3. (r3 work v3)(r3 type Manager)
4. (r4 manage v4)(r4 type Manager)
5. (r5 work v5)(r5 type Researcher)
6. (r6 manage v6)(r6 type Researcher)
7. (r7 work v7)(r7 type Director)
8. (r8 manage v8)(r8 type Director)
```

In these annotations, `ri` and `vi` are instances of concepts that are inferred from the properties definitions and from type restrictions that are included in the annotations. In (Luong and al. 2007), the effects of modifying the ontology on this set of annotations are studied. The original ontology is modified, deleting the concept `Director`, merging `PhdStudent` and `Trainee` into the concept `Student`. The new ontology that is obtained is defined below:

```
Concepts: Person, Student, Researcher, Director, Team, Project;
Researcher is-a Person;
Director is-a Person;
Student is-a Person;

Properties: work, manage;
Person work Team;
Director manage Project;
```

A consequence of the modifications of the ontology is that the annotations 2, 3, 4 and 6 become inconsistent.

3. CONVERTING AND ONTOLOGY TO A TYPE SYSTEM

We can make a parallel between ontologies and semantic annotations with programs that are also formal systems. Concepts can be viewed as types and subsumptions as type inclusions. In this context, properties get signatures that define their domains and co-domains.

The initial ontology given in section 2. can be described as follows. The sign `<=` denotes type inclusion.

```
Person, PhdStudent, Trainee, Manager, Researcher, Director, Team,
    Project : type;

PhdStudent <= Person;
Trainee <= Person;
Manager <= Person;
Researcher <= Manager;
Director <= Manager;

work : Person ->  Team;
manage : Manager -> Project;
```

The semantic annotations can be seen as expressions in a programming language. Instances are represented by objects (let's say constants). Depending of the programming language, the type of objects can be inferred or declared. We prefer to declare them as it is the case in languages with strong typing to optimize type verification and obtain as much error messages as possible.

```
r1 : Person;
r2 : PhdStudent;
r3, r4 : Manager;
r5, r6 : Researcher;
r7, r8 : Director;
v1, v2, v3, v5, v7  : Team;
v4, v6, v8 : Project;
```

If we apply the same modifications to the ontology as previously, the type system looks now as follows:

```
Person, Student, Researcher, Director, Team, Project : type;

Student <= Person;
Researcher <= Person;
Director <= Person;

work : Person -> Team;
manage : Director -> Project;
```

Applying a traditional type checker to our set of semantic annotations, will produce error messages.

In the following declarations

```
r2 : PhdStudent;
r3, r4 : Manager;
```

the types (concepts) `PhdStudent` and `Manager` are not declared and these declarations are not legal.

In the annotations 2, 3, and 4, `r2`, `r3,` and `r4` are not of type `Person` as declared by the signatures of the properties `work` and `manage`. In the annotation 6, `r6` is not of type `Director`.

(Klein 2002) shown that modifying a part of an ontology can imply some inconsistencies somewhere else in this ontology or when it is used. (Luong 2007) proposed some rules to detect these inconsistencies. Viewing an ontology as a type system, we can say that checking the consistency of ontologies and annotations can be done by a traditional type-checking.

4. CONCLUSION

In many aspects, information systems can be compared to programs. In particular, ontologies used in these information systems can be compared to the declarative parts of programs, and more precisely to type systems. In this paper, we have shown this fact on a small example, showing that checking the consistency of an ontology can be done by type-checking. Type systems have been heavily studied in the context of programming languages semantics, defining some notion as overloading, polymorphism, type parameters, etc. Modern programming languages take advantage of this in term of modularity. Ontologies definition formalisms use a lot of notion coming from the logic world. We think that a fertilization coming from the types world should be able to design more powerful and more modular ontologies definition languages.

REFERENCES

Book

Gunter C. A., 1992. *Semantics of Programming Languages*. MIT Press, Cambridge MA, USA.

Thesis

Luong P.-H., 2007. *Gestion de l'évolution d'un Web sémantique*. PhD Thesis, Ecoles des Mines, Paris, France.

Journal

Berners-Lee T. et al, 2001. The Semantic Web. *In Scientific American*, May 2001.

Horrocks I. et al, 2003. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *In Journal of Web Semantics*, 1(1), p. 7-26.

Gruber T. R., 1993. A Translation Approach to Portable Ontology Specifications. *In Knowledge Acquisition*, 5(2), p. 199-220.

Luong P.-H. and Dieng-Kuntz R., 2007. A rule-based approach for semantic annotation evolution. *In The Computational Intelligence Journal, 23(3),* p. 320-338.

Pressman et al., 1998. Can internet-based application be engineered ? IEEE Software, 15(5), p. 104-110.

Conference paper

Despeyroux Th., 2004, Practical semantic analysis of Web sites and documents. *In Proceedings of the 13th World Wide Web Conference (WWW'2004), New-York City, USA*.

Fensel D. et al., 1998, Ontobroker: the Very High Idea. *In Proceedings of the 11th International FLAIRS Conference (FLAIRS-98), Sanibel Island, Florida*.

Grosof B. N. et al., 2003. Description Logic Programs : Combining Logic Programs with Description Logic. *In Proceedings of World Wide Web Conference (WWW'2003), Budapest, Hungary*.

Klein M. et al., 2003. Ontology versioning and change detection on the. *In Proceedings of the 13rd European conference on knowledge engineering and knowledge management (EKAW'2002), Siguenza, Spain.*

Luong P.-H. et al., 2007. Evolution de l'ontologie et gestion des annotations sémantiques inconsistantes. *In Proceedings of Extraction et gestion des connaissances (EGC'2007), Namur, Belgique*.

Luong P.-H. et al., 2006. Managing semantic annotations evolution in the coswen. *In Proceedings of the Third National Symposium on Research, Development and Application of Information and Communication Technology (ICT.rda'06), Quebec City, Canada*.

Murugesan et al, 2001. Web engineering : A new discipline for development of web-based systems. *In Proceedings of the Web engineering, Software Engineering and Web Application Development.*

Patel-Schneider P. F. and Horrocks I., 2006. Position Paper: A Comparaison of Two Modelling Paradigms in the Semantic Web. *In Proceedings of the World Wide Web Conference (WWW'2006), Edinburgh, UK.*

Weithöner T. et al, 2004. Efficient Processing of Huge Ontologies in Logic and Relational Databases. *In Proceedings of Ontologies, Databases, and Applications of Semantics Conference (ODBASE'2004), Larnaca, Cyprus, On the Move to Meanongful Internet System, LNCS 3292*

Technical Report

Fensel D. et al., 1998, On2broker: Lessons Learned from Applying AI to the Web. *Technical report, Institute AIFB*

Others

Berners-Lee T., 1998, A Road Map to the Semantic Web, http://www.w3.org/DesignIssues/Semantic.html.

Klyne G. and Carrroll J., 2004, Resource description framework(rdf) : Concepts and abstract syntax, W3C Recommendation, http://www.w3.org/TR/rdf-concepts/.