On the Refinement of Ontologies

Grigoris Antoniou[1,2] and Athanasios Kehagias[3,4]

[1] School of CIT, Griffith Univ., Australia
[2] Dept. of Applied Informatics, Univ. of Macedonia, Greece

[3] Dept. of EE and CS, Aristotle Univ., Greece
[4] Dept. of Mathematics and CS, American College of Thessaloniki, Greece

Abstract. Ontologies have emerged as one of the key issues in information integration and interoperability and their application to knowledge management and electronic commerce. A trend towards formal methods for ontology management is obvious. This paper discusses a concept which can be expected to be of great importance to formal ontology management, and which is well-known in traditional software development: the refinement of ontologies. We define and discuss the concept, give illustrating examples, and highlight its advantages as compared to other forms of ontology revision.

1 Introduction

An ontology defines the terminology of a domain: it describes the concepts that constitute the domain, and the relationships between those concepts. Any information system uses its own ontology, either implicitly or explicitly. As applications become increasingly complex we can observe a trend towards the explicit representation and management of ontologies.

Ontologies have emerged as one of the key issues in the integration of information and the interoperability of systems, and their application to knowledge management and electronic commerce (see, for example, [13]). The existence of a common set of definitions of terminology, a common ontology, makes the interoperation of different information systems much easier. This is the approach takes by many researchers in information integration, e.g. [5, 9, 11, 15]. Moreover, ontologies are useful in information retrieval, where the use of the right keywords is critical for the successful processing of a user query [7].

Ontologies need to be developed and maintained just like other software parts or knowledge bases. In fact, these tasks are quite complex for ontologies. Firstly, there are various levels and domains in which terminology must be defined. For example, [6] distinguishes between top-level ontologies (general concepts of space, time, actions etc.), domain ontologies (concepts in a specific domain), task ontologies (concepts of a specific type of problems), and application ontologies (concepts of a specific application, depending on both task and domain concepts).

And secondly, ontologies are not developed to remain stable, but are subject to continual *change*. Change is caused by several factors: better ways of organising information are discovered; new concepts evolve over time and need to be added to the ontology; in information integration, the addition of new information resources may result in changes to the common ontology. Consulting companies and retailers, to mention just two business sectors, are known to spend large amounts of money to update their ontologies [12, 13].

This paper will study a particular kind of change, namely *refinement* (and its opposite process, *abstraction*). These concepts are known from the area of software engineering, together with the underlying concept of a conservative extension, where they play a prominent role [16]. Conservativity of extension guarantees that the details added in the development process of software specifications are "non-corrupting" (the exact meaning will be discussed in a subsequent section). More pertinent for modern software systems is the fact that the conservative extension criterion also guarantees the integrity of modularization and the transparency of objects in the object-oriented framework.

The aim of this paper is to discuss the notion of a conservative extension, and its derivatives refinement and abstraction, in the context of ontologies. We will define and discuss these concepts, give illustrating examples, and discuss its advantages when compared to other forms of ontology revision. One particular advantage is locality of change in an information integration scenario, as will be explained later on.

2 On Ontologies

In philosophy, ontology is the study of existence, or a description of what exists [3]. In information systems and artificial intelligence, the word ontology refers to the terminology of a system or, to be more precise, an explicit specification of a conceptualisation: "the objects, concepts and entities that are assumed to exist in some area of interest, and the relationships that hold among them" [8].

As a minimum, an ontology will define taxonomic relationships (such as: "a professor is a staff member") and some other constraints on terms (such as: "every student must be enrolled in at least one course"). As [17] points out, while the purpose of an ontology is to define terminology, the form of an ontology is that of a knowledge base or database conceptual schema. Since ontologies are usually represented in logic-flavoured languages (such as KIF [4] and CycL [10]), in this paper we will often view an ontology as a first order theory.

It is instructive to see how taxonomic information can be represented in logic. Consider the tree in Figure 1 (all figures are presented at the end of this paper) which represents classes of staff in the School of CIT at Griffith University. Its logical representation consists of the following logical rules.

$$
\begin{aligned}
&faculty(X) \rightarrow staff(X)\\
&adminStaff(X) \rightarrow staff(X)\\
&researchStaff(X) \rightarrow staff(X)\\
&visitingStaff(X) \rightarrow staff(X)
\end{aligned}
$$

3 Conservative Extensions

The formal specification of software has been instrumental in improving the reliability of programs and our understanding of the implementation process itself. The specification language Z [14] is typical of a number of languages that have been developed for this purpose. An important aspect of this methodology is its view of what it means for one specification S' to be a refinement of another S. One interpretation of this relationship, for instance, is that S' is an implementation of S, so S' "has more detail" than S. Influential work by Turski and Maibaum [16] advocated the idea that this be modelled in two steps: (i) regard S and S' as logical theories, and (ii) let S' be a conservative extension of S.

Formally we say that S' (where $S \subseteq S'$) is a conservative extension of S if the formulae that follow from S' and can be expressed in the language of S are exactly those which follow from S. Intuitively, one can see that this captures the idea that while S' has more detail than S, the details are "inessential" or "non-corrupting". More pertinent for modern software systems is the fact that this conservative extension criterion, as argued in [16], also guarantees the integrity of modularization and the transparency of objects in the object-oriented framework.

These ideas carry over easily to ontologies. Let O and O' be two ontologies. Then O' is a *conservative extension* of O iff, for every relevant φ statement in the terminology of O, φ follows from O iff φ follows from O'. This formulation has two parameters: relevant statements and a notion of "follows"; both depend on the underlying ontological representation language. For example, if one just uses concept trees, then the relevant statements have the form $t \rightarrow t'$ (for terms t and t'), and "follows" is based on a check whether t' is an ancestor of t. In case the ontologies are viewed as logical theories, extensions are defined exactly as for logical software specifications, as described above.

We say that O' is a *refinement* of O iff O' is a conservative extension of O. In this case we say that O is an *abstraction* of O'.

Here are two examples where concepts organised as taxonomies are refined. Figure 2 shows a refinement of Figure 1 through the addition of a new concept "technical support staff":

In Figure 3 new concepts "academic" and "non-academic" are introduced at the appropriate level. It is a refinement of Figure 2 because no new relationships between the concepts of Figure 2 are introduced, and none are lost.

Not every extension leads to a refinement. Figure 4 shows an ontology which is not a conservative extension of Figure 1 because it introduces a new relationship between old concepts (faculty and visiting staff). Although faculty bears the same name in both ontologies, their meaning is different, since in Figure 4 it includes the visiting staff, too.

4 Ontology Refinement in Information Integration

4.1 Information Integration

Information integration (II) [2, 5, 9, 11, 15] seeks to bridge the gap between the user needs and the information available from different sources by introducing a meta information source, called *mediator* or *facilitator*, which provides in an integrated fashion access to information from various sources. The basic architecture of the information integration tools is usually agent-based. The three main types of agents reflect the main parts in the information integration scenario: (i) *user agent*, which collects and passes on the user requests; (ii) *database agent* (or information source agent or wrapper), which makes available the services of a database management system (or an information/knowledge source); and (iii) *mediator* or *facilitator* whose aims are to transform the user request to appropriate queries to database agents, to collect and integrate the responses, and to pass on an appropriate response to the user agent.

Each agent may use its own ontology. Let O_i denote the ontology of the i-th database agent, U_i denote the ontology of the i-th user agent, and CO denote the commmon ontology used by the mediator. To avoid clashes between symbols that may bear different meanings we assume that these ontologies are mutually disjoint. This is not uncommon in practical applications, and is easily achieved by the addition of a prefix to the concept names (for example, the term *staff member* in the common ontology CO is denote as "CO-staff member").

4.2 Ontology translation

One of the basic function of information integration is to translate the user query to the common ontology, and from there to the ontologies of the information resources. Thus ontology translation is central to information integration. Here we give a formal definition of this concept.

Given two ontologies O_1 and O_2 (represented as first order theories), a translation from O_1 to O_2 is a first order theory T which contains definitions of the concepts of O_1 in terms of O_2.

Here is an example. Let O_1 be the ontology in Figure 5 and O_2 the ontology in Figure 6. Then the following set T defines a sound translation from O_1 to O_2.

$$O_1\text{-}staff(X) \leftrightarrow O_2\text{-}staff(X)$$
$$O_1\text{-}faculty(X) \leftrightarrow O_2\text{-}faculty(X) \vee O_2\text{-}visitingStaff(X)$$
$$O_1\text{-}resStaff(X) \leftrightarrow O_2\text{-}resStaff(X)$$
$$O_1\text{-}adminStaff(X) \leftrightarrow O_2\text{-}adminStaff(X)$$

The requirement we impose on T is that $O_2 \cup T$ is a conservative extension of O_2. In other words, T extends the theory O_2 without changing the meaning of any symbol in O_2. The example above satisfies this condition.

While many translations may be defined from the syntactic point of view, not all translations will be *semantically sound*. For a translation T to be sound, it needs to define the concepts of O_1 in such a way that the relationships specified in O_1 hold for the translation, too. Logically this means that O_2 together with T should entail O_1: $O_2 \cup T \models O_1$.

The example above defined a semantically sound translation. Here is an example of a translation that is not sound. Consider the ontology O_1 as specified in Figure 7, and the following translation T:

$O_1\text{-}staff(X) \leftrightarrow O_2\text{-}staff(X)$
$O_1\text{-}faculty(X) \leftrightarrow O_2\text{-}faculty(X)$
$O_1\text{-}visitingStaff(X) \leftrightarrow O_2\text{-}visitingStaff(X)$

T is not a sound translation from O_1 to O_2 because the relationship between $O_1\text{-}faculty$ and $O_1\text{-}visitingStaff$ is not reflected. Obviously T should have defined *faculty* in O_1 to be *academic staff* in O_2.

4.3 Refinement and Locality of Change

Changes to ontologies pose a difficult problem in information integration, since there is, in general, no central control over the database agents. If the organisation responsible for a database agent decides to change its ontology, then the II system is faced with a dilemma: either it decides to remove the agent from the system (thus reducing its functionality and utility), or it must incorporate the changes into the entire system. In the following we discuss the latter case.

Suppose that the common ontology is CO, and a database agent wishes to change its ontology from O_i to O_i'. Also, let T be a sound translator from O_i to CO. Obviously T will need to be changed. But it may be the case that CO is not expressive enough to incorporate the new ontology O_i', so CO may have to be changed as well. In the worst case, this may lead to necessary changes for all translators T_j with $i \neq j$.

It is here that the advantage of refinement becomes apparent. We can show that if the change from O_i to O_i' is conservative, then the change remains local: even in the worst case, only T and CO need to be refined. It is not difficult to see this. We formulate the argument using two formal results.

Theorem 1 *Let T be a sound translation from ontology O to an ontology CO. Further suppose that O' is a conservative extension of O. Then there are conservative extensions CO' and T' of CO and T, such that T' is a sound translator from O' to CO'.*

The theorem follows immediately from the following observation. For $A \models B$ and a refinement B' of B, $A \cup (B' - B) \models B$. Moreover, due to our assumption about the disjointness of terminologies from different ontologies in the information integration scenario (see last paragraph of subsection 4.1), $A \cup (B' - B)$ is a refinement of A, since $(B' - B)$ cannot introduce any new relationships among symbols of A.

Theorem 2 *Let T be a sound translator from O to CO, and let CO' be a refinement of CO. Then T is also a sound translator from O to CO'.*

The proof is obvious. The first result says that even if we have to change the common ontology CO to accommodate the change in the ontology O_i of a database agent, this change needs only be a refinement. In that case, the translators for the other database agents need not be changed, as the second result shows.

Here is an example. Let CO be the ontology of Figure 6, and O_1 the ontology of Figure 5. Also, suppose that the translator T_1 is the translator given in subsection 4.2. CO represents the general organisation of a department, while O_1 reflects the organisation of the School of CIT at Griffith University. Further suppose that there is also an ontology O_3, same as O_1, which reflects the organisation of the computer science department at the University of Albany (of course, we use highly simplified ontologies due to space limitations, just to illustrate our argument), and an associated translator T_3.

Now suppose that Griffith University decides that the academic ranks should be included in the ontology of the CIT agent. Thus leaves *lecturer, seniorLecturer, associateProfessor, professor* are added as sons of the *faculty* node, and give us ontology O'_1. The common ontology CO in its current form is unable to represent this information, so it must be extended, too. So we might decide to add the nodes *assistantProfessor, associateProfessor, professor, distinguishedProfessor* as nodes of *faculty* (in CO). This gives us the new common ontology CO'. Finally we extend T_1 to T'_1 by adding the following translation rules:

$$O'_1\text{-}lecturer(X) \leftrightarrow CO'\text{-}assistantProfessor(X)$$
$$O'_1\text{-}seniorLecturer(X) \leftrightarrow CO'\text{-}associateProfessor(X)$$
$$O'_1\text{-}associateProfessor(X) \leftrightarrow CO'\text{-}Professor(X)$$
$$O'_1\text{-}professor(X) \leftrightarrow CO'\text{-}distinguishedProfessor(X)$$

It is not difficult to see that T'_1 is a sound translator from O to CO'. Moreover, ontology O'_1 and CO' are refinements of O_1 and CO respectively. The translator T_3 need not be modified.

5 Conclusion

We discussed the notion of a *conservative extension*, and its derivatives: *refinement* and *abstraction*, in the context of ontologies. We defined these concepts, gave illustrating examples, and showed that refinement allows for local changes in information integration, as opposed to other forms of ontology revision.

Several ontological issues can be discussed within the framework of this paper. For example, we are currently exploring the introduction of a *distance* between ontologies. This will be useful in selecting the "nearest" (subject to certain constraints) refinement of an ontology or the "least common refinement" of two ontologies. There is a connection with research on belief revision where the notion of *minimal change* is central [1]. The distance function we are currently investigating is obtained from *graph-theoretic* considerations (it is easy to show that certain logical languages can be corresponded to directed graphs). The graph-based approach may also facilitate the development of efficient algorithms, for example to test whether an ontology is a refinement of another or to find the "least common refinement" of two ontologies.

References

1. C.E. Alchourron, P. Gardenfors and D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic* 50 (1985), 510-530.
2. V.R. Benjamins, S. Decker, D. Fensel, E. Motta, E. Plaza, G. Schreiber, R. Studer and B. Wielinga. IBROW – An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web. In *Proc. ECAI'98 Workshop on Applications of Ontologies and Problem Solving Methods.*
3. A. Flew. *A Dictionary of Philosophy*, p. 238. St Martin's Press, New York 1979.
4. M.R. Genesereth and R.E. Fikes. *Knowledge Interchange Format, Version 3.0, Reference Manual.* Technical Report Logic-92-1, Computer Science Department, Stanford University 1992.
5. M.R. Genesereth, A.M. Keller and O.M. Duschka, O. Infomaster: An Information Integration System. In *Proc. 1997 ACM SIGMOD Conference*, ACM Press 1997.
6. N. Guarino. Formal Ontology and Information Systems. In N. Guarino(ed.), *Formal Ontology in Information Systems*, IOS Press 1998, 3-15.
7. N. Guarino. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction and Integration. In *Lecture Notes in Computer Science* 1299, Springer, 1997, 139-170.
8. T.R. Gruber. Towards Principles for The Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli (Eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kluwer 1995.
9. C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, P.J. Modi, I. Muslea, A.G. Philpot and S. Tejada. Modeling Web Sources for Information Integration. In *Proc. AAAI-98.*
10. D.B. Lenat and R.V. Guha. *Building Large Knowledge-based Systems*, Addison-Wesley 1990.
11. A.Y. Levy, A. Rajaraman and J.J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. 22th International Conference on Very Large Data Bases*, 1996.
12. F. Maruyama. Personal communication, Fujitsu 1998.
13. D. O'Leary. Using AI in Knowledge Management: Knowledge Bases and Ontologies. *IEEE Intelligent Systems* 13,1 (1998):34-39.
14. J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall 1989.
15. M. Tork Roth and P. Schwarz. Don't Scrap it, Wrap it! A Wrapper Architecture for Legacy Data Sources. In *Proc. 23rd International Conference on Very Large Data Bases*, 1997.
16. W.M. Turski and T.S.E. Maibaum. *The Specification of Computer Programs.* Addison-Wesley 1987.
17. A. Waterson and A. Preece. Verifying Ontological Commitment in Knowledge-Based Systems. In *Proc. AAAI-97 Workshop on the Verification and Validation of Knowledge-Based Systems*, AAAI Press 1997. Extended version accepted for publication by *Knowledge-Based Systems.*