MIT Sloan School of Management

MIT Sloan Working Paper 4621-06
Composite Information Systems Laboratory (CISL) Working Paper 2006-06
June 2006

A Lightweight Ontology Approach to Scalable Interoperability

Hongwei Zhu, Stuart Madnick

A Lightweight Ontology Approach to Scalable Interoperability

Hongwei Zhu , Stuart Madnick
Massachusetts Institute of Technology
Cambridge, MA USA
{mrzhu, smadnick}@mit.edu

Working Paper CISL# 2006-06

June 2006

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E53-320
Massachusetts Institute of Technology
Cambridge, MA 02142

A Lightweight Ontology Approach to Scalable Interoperability

Hongwei Zhu[1], Stuart E. Madnick[1]

[1] Massachusetts Institute of Technology,
30 Wadsworth Street, E53-320, Cambridge, MA 02142, USA
{mrzhu, smadnick}@mit.edu

Abstract. There are many different kinds of ontologies used for different purposes in modern computing. Lightweight ontologies are easy to create, but difficult to deploy; formal ontolgies are relatively easy to deploy, but difficult to create. This paper presents an approach that combines the strengths and avoids the weaknesses of lightweight and formal ontologies. In this approach, the ontology includes only high level concepts; subtle differences in the interpretation of the concepts are captured as context descriptions outside the ontology. The resulting ontology is simple, thus it is easy to create. The context descriptions facilitate data conversion composition, which leads to a scalable solution to semantic interoperability among disparate data sources and contexts.

Keywords: lightweight ontology, context, mediation, scalability.

1 Introduction

Ontologies have been widely used in modern computing for purposes such as communication, computational inference, and knowledge organization and reuse [5]. For different purposes, there are a variety of different ontologies that range from a glossary, to a taxonomy, a database schema, or a full-fledged logic theory that consists of concepts, relationships, constraints, axioms, and inference machinery. As illustrated in [14], a variety of ontologies form a continuum from lightweight, rather informal, to heavyweight, and formal ontologies.

The lightweight ontology approach and the formal ontology approach are often used differently and have different strengths and weaknesses. Lightweight ontologies usually are taxonomies, which consist of a set of concepts (i.e., terms, or atomic types) and hierarchical relationships among the concepts. It is relatively easy to construct a lightweight ontology. To use a lightweight ontology for interoperability purposes, all parties need to agree on the exact meaning of the concepts. Reaching such agreements can be difficult. The lightweight ontology and the agreements together form a standard that all parties uniformly adopt and implement. That is, a lightweight ontology is often used to support strict data standardization. In contrast, a formal ontology uses axioms to explicitly represent subtleties and has inference capabilities. It can support data standardization in a different way, that is, the agreements are explicitly specified in the ontology. More often, a formal ontology is

used to allow for data heterogeneity and to support interoperability, in which case the different interpretations and representations of data are explicitly captured in the ontology. In either case, a formal ontology disambiguates all concepts involved. Once created, a formal ontology can be relatively easy to use. But it often takes tremendous effort to create a formal ontology due to the level of detail and complexity required.

To summarize, lightweight ontologies are often used as data standards; as artifacts, they are simple, and thus easy to create, but difficult to use. Formal ontologies are often used to support interoperability of heterogeneous data sources and receivers; as artifacts, they are complex and difficult to create, but easy to use. Either approach has its weakness that limits its effectiveness.

It is desirable to have an approach that combines the strengths and avoid the weaknesses of the two ontological approaches. In the paper, we present such an approach, which is developed in the COntext INterchange (COIN) project [2, 3, 18] for semantic data interoperation purposes. It uses a lightweight ontology, which provides the structure for organizing context descriptions to account for the subtleties of the concepts in the ontology. We will use the terms *COIN ontology* and *COIN lightweight ontology* interchangeably. COIN also implements a reasoning algorithm to determine and reconcile semantic differences between different data sources and receivers.

The rest of the paper is organized as follows. In Section 2, we describe the COIN lightweight ontology approach. In Section 3, we present the scalability benefit of the approach. In Section 4, we discuss related work. In Section 5, we conclude and point out future research.

2 COIN Lightweight Ontology

We will use an online price comparison example to illustrate the COIN lightweight ontology approach.

2.1 Online Price Comparison Example

Numerous vendors make their pricing information available online. With web wrappers [1] and the increasing adoption of web services, one can gather price data and compare offers from different vendors. To perform meaningful comparisons, one has to reconcile the semantic differences of price data, especially when data is from vendors scattered around the world [16].

Consider a scenario where data is from 30 vendors from 10 different countries. For simplicity of discussion in this paper, let us assume that all vendors quote prices using the same schema and same *Product* identification, represented using the following first order predicate:

quote(*Product, Price, Date*)

but different vendors use different conventions so that the price values are interpreted differently depending on which vendor provides the quote. Table 1 provides a few examples of different interpretations of price.

Table 1. Interpretations of Price.

Vendor	Interpretation of Price
1	In 1's of USD, taxes and S&H included
2	In 1's of USD, taxes and S&H excluded
3	In thousands of Korean won, taxes and S&H excluded
…	…
30	In millions of Turkish lira, taxes included

Let us assume that each vendor uses a different convention, thus we have 30 unique conventions, which we call *contexts*. We can label vendor *i*'s context as c_i. For simplicity, we will assume that users normally adopt a vendor context. In this scenario, to allow users in all contexts to meaningfully compare vendor prices, it is necessary that price data from other contexts be converted to the user context, which would require 870 (i.e., 30*29=870) conversions. Hand-coding these conversions and maintaining them over time, since contexts do change, can be costly and error-prone.

2.2 COIN Lightweight Ontology

In the example, there are many subtle differences in the meaning of the generic concept *price*. It is important that these subtleties are captured and the differences are reconciled for meaningful comparisons.

In addition to *is_a* relationship, the COIN ontology also includes *attribute* as a binary relationship between a pair of concepts. Attributes are also called roles, and correspondingly attribute names are called role names. For example, *price* can be the *hasPrice* attribute of *product*. Conversely, *product* can be the *priceOf* attribute of *price*. To capture the subtle differences in meaning, we introduce *modifier* as a special kind of attribute. The values of modifiers are specified as context descriptions outside the ontology. Fig. 1 shows a graphic representation of the COIN lightweight ontology for the online price comparison example.

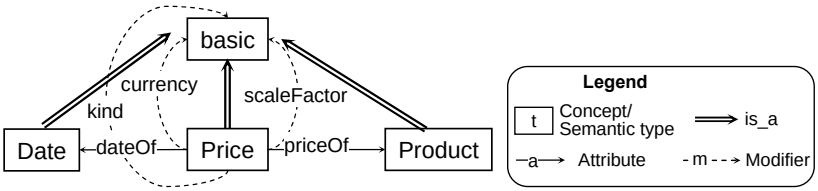

Fig. 1. COIN lightweight ontology for online price comparison example.

In this ontology, we include *basic* as the most generic type, which is similar to *thing* as the root in many object-oriented models. We include three modifiers: *kind*, *currency*, and *scaleFactor*. Each modifier captures a particular aspect in which the

underlying concept can have different interpretations. Contexts are described by assigning values to modifiers present in the ontology. For example, context c_2 (refer to vendor 2 in Table 1) can be represented by a set of <*modifier*, *value*> pairs:

c_2 := { <*kind*, *basePrice*>,
<*currency*, *usd*>,
<*scaleFactor*, 1> }

2.3 Characteristics of COIN Lightweight Ontology

A COIN ontology, as shown in Fig. 1, includes only high level concepts (plus the context modifiers). Thus it is simple and relatively easy to create and reach agreement. But the involved parties do not need to agree on the details of each concept. Each party can continue to use its preferred interpretation for each high level concept. In other words, each party can *conceptually* have its own local ontology. Fig. 2 depicts the conceptual local ontologies for vendors 2 and 3. To avoid clutter, we have omitted attribute names in the figure.

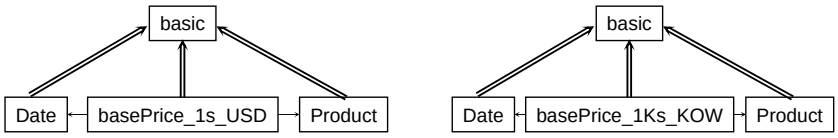

Fig. 2. Conceptual local ontologies for vendor 2 (left) and vendor 3(right), derivable from COIN lightweight ontology shown in Fig. 1.

These local ontologies are not part of the COIN lightweight ontology, but they can be derived from the COIN ontology using the context descriptions. In other words, the COIN lightweight ontology provides a structured way to describe contexts and derive refined local ontologies.

Furthermore, a more traditional global ontology that integrates all the local ontologies could be constructed from the COIN ontology and the accompanying context descriptions. A graphic representation of such a global ontology for the online price comparison example is given in Fig. 3, which includes two intermediate layers (i.e., the layers starting with *BasePrice* and *In USD* concepts, respectively). Concepts in each layer remove a certain kind of ambiguity. For example, *BasePrice* indicates the kind of price, which does not include shipping and handling charges. The nodes below it further refine the base price concept by specifying the currency, e.g., in USD. Alternatively, the intermediate layers can be omitted. In this case, specialized concepts on the leaf level, such as *basePrice_1s_USD*, directly connect to the generic *Price* concept.

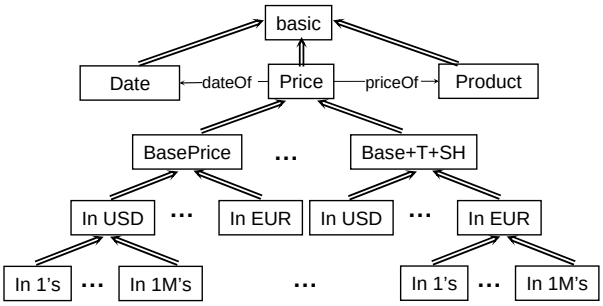

Fig. 3. An example fully-specified global ontology for the online price comparison example.

Ontologies are design artifacts. Comparing the artifacts shown in Fig. 1 and Fig. 3, we observe that the COIN approach creates much simpler ontologies – though, for many purposes, they are functionally equivalent. As discussed in [10, 18], the COIN approach has several advantages. First, the COIN ontology is usually much simpler, thus easier to manage. Although in practice it is unlikely that one would create an ontology to include all possible variations (e.g., *basePrice_1M's_USD*), a COIN ontology is still much easier to create than any ontology similar to the one in Fig. 3 even with a smaller number of refined concepts. Second, related to the first point, although the COIN ontology is simple, it provides the means to derive all refined concepts as illustrated in Fig. 3. Third, a COIN ontology facilitates consensus development, because it is relatively easier to agree on a small set of high level concepts than to agree on every piece of detail of a large set of fine-grained concepts. And more importantly, the COIN ontology is much more adaptable to changes. For example, when a new concept "base price + S&H in 1000's of South Korean Won" is needed, the fully specified ontology may need to be updated with insertions of new nodes. The update requires the approval of all parties who agreed on the initial ontology. In contrast, the COIN approach can accommodate this new concept by adding new context descriptions without changing the ontology.

3 Scalable Interoperability with COIN Lightweight Ontology

When data sources and data receivers are in different contexts, conversions, also called lifting rules, are needed to convert data from source contexts to the receiver context. We call the set of conversions from a context to another context a *composite conversion*. When conversions are specified pair-wise between contexts, it requires $\sim n^2$ composite conversions to achieve interoperability among n contexts. It is costly and error-prone to develop and maintain such a large number of conversions. Thus approaches that hard-code the $\sim n^2$ composite conversions do not scale well when n increases.

The use of lightweight ontology in COIN makes it possible to avoid the above mentioned problem. In addition to using ontology and contexts to represent semantic heterogeneity, COIN also has a reasoning component to determine and reconcile

semantic differences. We explain how COIN achieves scalability though conversion composition in the remainder of the section.

3.1 Conversion Composition

In COIN, conversions are specified for each modifier between different modifier values. For example, a conversion can be defined for *currency* modifier to convert values in different currencies such as by using an exchange rate function represented by the following predicate:

r(*CurFrom*, *CurTo*, *Day*, *Rate*)

It returns an exchange *Rate* from *CurFrom* currency to *CurTo* currency on a given *Day*. The function can be implemented externally as a table lookup or as a callable service[1]. We call a conversion defined for a single modifier a *component conversion*.

Once all component conversions are defined, composite conversions can be composed automatically using a context reasoning algorithm. Fig. 4 illustrates the concept of conversion composition.

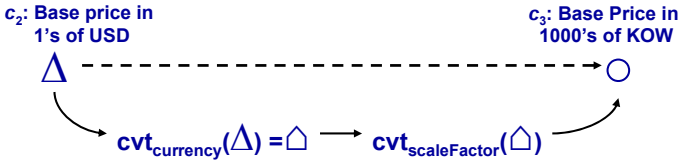

Fig. 4. Composite conversion composed using component conversions.

In Fig. 4, the triangle symbol on the left represents the price data in context c_2, i.e., base price in 1's of USD; and the circle symbol on the right represents the price data in context c_3, i.e., base price in 1000's of Korean won (KOW). For data in context c_2 to be viewed in context c_3, they need to be appropriately converted by applying the appropriate composite conversion. The dashed straight arrow represents the application of the composite conversion.

With the COIN lightweight ontology approach, the composite conversion can be automatically composed using the predefined component conversions. As shown in Fig. 4, we first apply the component conversion for *currency* modifier (represented by $cvt_{currency}$), then apply the component conversion for *scaleFactor* modifier (represented by $cvt_{scaleFactor}$).

The composition algorithm, shown in Fig. 5, is quite simple. In COIN project, it is implemented using abductive constraint logic programming (ACLP) [8].

[1] In many applications using COIN, such conversion functions are implemented by use of web-wrapped services, such as the www.oanda.com currency conversion web site.

Input: data value V, corresponding concept C in ontology,
source context C1, target context C2
Output: data value V (interpretable in context C2)

```
Find all modifiers of C
  For each modifier mi
       Find and compare mi's values in C1 and C2
       If different: V=cvt_mi(V); else, V=V
Return V
```

Fig. 5. Algorithm for composing composite conversion using component conversions.

3.2 Scalability Benefit

The primary benefit of the composition capability is the small number of component conversions required, thus increased scalability when many data sources and contexts are involved in data integration applications [17, 18].

In the worst case, the number of component conversions required by the light-weight ontology approach of COIN is:

$$\sum_{i=1}^{m} n_i (n_i - 1)$$

where n_i is the number of unique values that the i^{th} modifier has to represent all contexts, m is the number of modifiers in the light-weight ontology.

Symbolic equation solver techniques have been used in COIN to exploit the relationships between component conversions of a same modifier [2]. These techniques further reduce the number of component conversions needed.

In many cases, the component conversion for a modifier can be parameterized, i.e., the component conversion can be applied to convert for any given pair of modifier values. In this case, we only need to supply one component conversion for the modifier, regardless of the number of unique values that the modifier may have. The exchange rate function given earlier is such an example; with it, we only need one component conversion for the *currency* modifier.

Let us use the online price comparison example to illustrate the scalability benefit of the approach. With the given scenario, we can model the 30 unique contexts using the three modifiers in the light-weight ontology shown in Fig. 1. Suppose the number of unique values of each modifier is as shown in Table 2.

Table 2. Modifier values.

Modifier	Unique values
currency	10, corresponding to 10 different currencies
scaleFactor	3, i.e., 1, 1000, 1 million
kind	3, i.e., base, base+tax, base+tax+S&H

In the worst case, the light-weight ontology approach needs 102 (i.e., 90+6+6) component conversions. Since the conversions for *currency* and *scaleFactor* modifiers are parameterizable, the actual number of component conversions needed is

further reduced to 8, which is a significant improvement from the 870 composite conversions required when conversions are specified pair-wise between contexts.

4 Related Work and Discussion

The most commonly cited definition for ontology is given in [4], where an ontology is a "formal explicit specification of a share conceptualization". But as discussed in [5, 14], there is not a consensus definition for ontology, and there are many types of ontologies, some of which use formal logic to explicitly capture the intended meanings, and others use a set of mutually agreed terms to provide a shared taxonomy. In the latter case, the intended meanings are not explicitly captured in the ontology, rather, they are implicitly captured in the agreement.

The term *lightweight ontology* has been used very loosely in the literature. Generally speaking, a lightweight ontology refers to a set of concepts organized in a hierarchy with *is_a* relationships. Data dictionaries, product catalogs, and topic maps are often considered to be lightweight ontologies. Opposite to lightweight ontologies are formal ontologies, which often use formal logic to specify constraints, relationships, and other rules that apply to the concepts [6, 11].

The use of ontology and contexts in the COIN approach is quite unique. The ontology provides the necessary structure for context descriptions; and the context descriptions, in turn, disambiguate the high level concepts in the ontology. The structure provided by the ontology also facilitates the provision of component conversions and the automatic composition of composite conversions necessary to enable semantic interoperability among contexts. The resulting solution is scalable because it requires significantly less manually created conversions.

There are other approaches that use ontology or contexts to enable interoperability among disparate data sources [15]. It is beyond the scope of this paper to provide a detailed comparison of these different approaches. We only make comments on a few approaches to further articulate the uniqueness of the COIN approach.

Contexts can be described without using an ontology. For example, they can be described using a context logic [12]. The so described contexts lack the structure like the one provided by the COIN ontology. As a result, a large number of conversions (i.e., lifting rules) are needed to enable semantic interoperability. A recent effort tries to categorize lifting rules and attempts to use the patterns revealed to devise general lifting rules [7]. More work is needed to show how these patterns help with creation of general lifting rules and how these rules can be applied to reason with multiple contexts.

Ontology is used in [13], where all types of data level and schema level heterogeneity in multiple data sources are explicitly represented using a semantic conflict resolution ontology (SCROL). For example, when acres and square meters are used in different sources to represent the *area* of a parcel of land, the SCROL ontology will explicitly represent the semantic difference by including two sub-concepts of area: *area_in_acre*, and *area_in_sq_meter*. A SCROL ontology resembles the one in Fig. 3. The ontology needs to be updated when a new kind of

heterogeneity is introduced, e.g., “area in square miles”. No characterization on the number of conversions needed is given in the paper.

Ontology is also used in [9] to provide structured context representation. However, we are not certain if their ontology would constitute a lightweight ontology. Nor does the paper provide an assessment about the number of conversions required.

5 Conclusion

The COIN lightweight ontology approach to semantic interoperability has several advantages. The ontology is simple, thus it is easy to create. The use of modifiers to capture subtle meaning differences provides the structure for describing the subtleties, and facilitates the provision of component conversions, with which any composite conversions can be composed dynamically to reconcile the semantic differences between the sources and the receivers of data.

For future research, we would like to explore the applicability of the COIN approach in other application domains, such as context-aware web services and peer-to-peer information sharing.

References

1. Firat, A., Madnick, S. E. and Siegel, M. D. (2000) "The Cameleon Web Wrapper Engine", *Workshop on Technologies for E-Services (TES'00),* Cairo, Egypt.
2. Firat, A. (2003) "Information Integration using Contextual Knowledge and Ontology Merging", PhD Thesis, Sloan School of Management, MIT.
3. Goh, C. H., Bressan, S., Madnick, S. and Siegel, M. (1999) "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information", *ACM TOIS,* **17**(3)**,** 270-293.
4. Gruber, T. R. (1993) "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition,* **5**(2)**,** 199-220.
5. Gruninger, M., Lee, J. (2002) “Ontology Applications and Design”, *Communications of the ACM*, **45**(2), 39-41.
6. Guarino, N. (1998) “Formal Ontology and Information Systems”, Proceedings of Formal Ontologies in Information Systems, IOS Press, 3-15.
7. Guha, R. and McCarthy, J. (2003) "Varieties of Contexts", *CONTEXT 2003***,** LNAI 2680, 164-177.
8. Kakas, A. C., Michael, A. and Mourlas, C. (2000) "ACLP: Abductive Constraint Logic Programming", *Journal of Logic Programming*, **44**(1-3)**,** 129-177.
9. Kashyap, V. and Sheth, A. P. (1996) "Semantic and Schematic Similarities between Database Objects: A Context-Based Approach", *VLDB Journal,* **5**(4)**,** 276-304.
10. Madnick, S. E. and Zhu, H. (forthcoming) "Improving data quality through effective use of data semantics", *Data & Knowledge Engineering*.
11. Mädsche, A. (2002) *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers.
12. McCarthy, J. and Buvac, S. (1997) "Formalizing Context (Expanded Notes)", In *Computing natural language* (Eds, Aliseda, A., van Glabbeek, R. and Westerstahl, D.), Sanford University.

13. Ram, S., Park, J. (2004) “Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflict”, *IEEE Transactions on Knowledge and Data Engineering*, **16**(2), 189-202.
14. Uschfold, M., Gruninger, M. (2004) “Ontologies and Semantics for Seamless Connectivity”,
15. Wache, H., Vögele, T., Visser, U., H. Stuckenschmidt, Schuster, G., Neumann, H. and Hübner, S. (2001) "Ontology-Based Integration of Information - A Survey of Existing Approaches", *IJCAI-01 Workshop: Ontologies and Information Sharing,* Seattle, WA**,** 108-117.
16. Zhu, H., Madnick, S. and Siegel, M. (2002) "Global Comparison Aggregation Services", *1st Workshop on E-Business,* Barcelona, Spain.
17. Zhu, H. and Madnick, S. E. (2004) "Context Interchange as a Scalable Solution to Interoperating Amongst Heterogeneous Dynamic Services", *3rd Workshop on eBusiness (WEB),* Washington, D.C., 150-161.
18. Zhu, H. (2005) "Effective Information Integration and Reutilization: Solutions to Technological Deficiency and Legal Uncertainty", PhD, Engineering Systems Division, MIT.