Structured Contexts with Lightweight Ontology

Hongwei Zhu, Stuart E. Madnick

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

Abstract

One of the challenges of dealing with multiple contexts is the significant effort required to provide all necessary lifting rules so that statements in one context can be viewed and understood in other contexts. In this paper, we introduce the notion of *structured contexts*, where a *lightweight ontology* is used to provide a structure for representing contexts. With structured contexts, specialized inference algorithms can be used to significantly reduce the number of lifting rules required. We use a semantic data integration example to illustrate the concept of structured contexts and the benefits of this novel use of lightweight ontology.

Introduction

Contexts play an important role in artificial intelligence [McCarthy, 1987] and many other areas of modern computing, such as information integration, process integration, and various applications of web services. One use of contexts is to allow information in one context to be viewed and understood in other contexts. In this case, *lifting rules* (also known as conversion rules) need to be supplied to convert information from its originating context to target contexts [McCarthy and Buvac, 1997]. When contexts are unstructured and lifting rules are supplied in a brute-force manner, the number of lifting rules can become too big to manage [Guha and McCarthy, 2003; Zhu, 2005].

To substantially reduce the number of lifting rules required, we introduce the notion of *structured contexts* and show its effectiveness using an information integration example. In our approach, we use a *lightweight ontology* to provide a structure for representing contexts, and use a specialized reasoner to compose complex lifting rules from a small set of pre-defined simple rules.

Example and Unstructured Contexts

In this section, we use an online price comparison example, where information from multiple sources needs to be integrated, to illustrate the problem with unstructured contexts.

Example

Numerous vendors make their pricing information online. With web wrappers [Firat et al., 2000] and the increasing adoption of web services, one can gather price data and compare offers from different vendors. To perform meaningful comparison, one has to reconcile the semantic differences of price data, especially when data is from vendors scattered around the world [Zhu et al., 2002].

Consider a scenario where data is from 30 vendors in 10 different countries. All vendors quote prices using the same schema, represented using the following first order predicate:

quote(*Product, Price, Date*)

but each uses a different convention 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 contexts. 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.

Lifting Rules between Unstructured Contexts

In the logic of context developed by McCarthy and Buvac [1997] and various other logics of context, contexts are described in two ways: one is by the collection of all formulae in a context; second is by the lifting rules that relate a formula in one context to a formula in another context.

For the first case, the formulae in the context of vendor *i* (labeled as context c_i) are the collection of ground atoms

quote(*Product*, *Price*, *Date*), which correspond to all the records of the vendor's quote database. These atoms together describe context c_i.

For the second case, the lifting rules often take one of the following two forms in the logic of context [McCarthy and Buvac, 1997]:

$$c_0 : \quad ist(c_i, p_i) \leftrightarrow ist(c_j, p_j).$$
$$c_0 : \quad ist(c_i, p_i) \rightarrow ist(c_j, p_j).$$

which establish the logical equivalence or logical implication relationships between formulae p_i and p_j in contexts c_i and c_j, respectively.

Suppose there exists an externally implemented function for exchange rates between a pair of currencies, represented using the following predicate:

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

The lifting rule to convert price from context c_2 to context c_3 (see Table 1) may look like the following:

$$c_0 : ist(c_2, quote(I, P, D)) \rightarrow ist(c_3, quote(I, X, D)), \\ r(usd, kow, D, R), X = P * R / 1000. \qquad (1)$$

The lifting rule to convert from c_3 to c_2 can be defined similarly.

When lifting rules are specified pair-wise for all contexts involved in our example of 30 vendors, we need 870 (i.e., 30*29) lifting rules to convert data between all the contexts. Since the number grows at a rate of n^2, the approach does not scale well when there exist a large number of unique contexts. In addition, various context logics are only semi-decidable even under a variety of restrictions [Guha et al., 2004].

With this approach, contexts are only implicitly described using ground atoms and a set of pair-wise defined lifting rules. In other words, this approach does not explicitly describe the characteristics of a context, e.g., price is quoted in a certain currency, with a certain scale factor, and with certain costs included or excluded. When these context characteristics are explicitly described in a well organized fashion, it is possible to exploit the structured descriptions to effectively solve the scalability problem.

Ontology and Structured Contexts

To explicitly describe characteristics of multiple contexts, we need a vocabulary interpreted consistently across all contexts. An ontology [Gruber, 1993] can supply such a vocabulary. In addition, an ontology can also provide a structure for representing contexts. Reasoning algorithms can be developed to exploit the structure of context representation to reduce the number of lifting rules required.

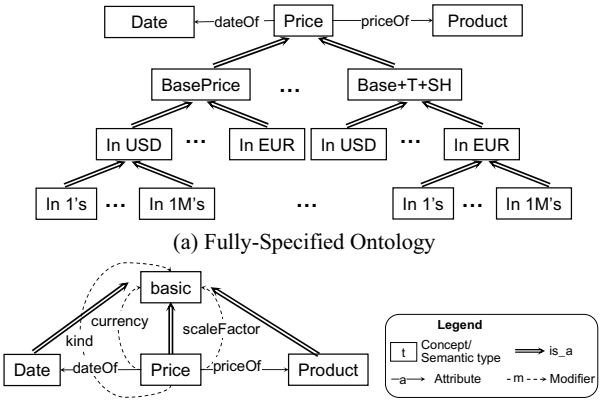
Fully-Specified vs. Lightweight Ontology

An ontology usually consists of a set of terms corresponding to a set of predefined concepts, relationships between concepts, and certain constraints. There are two types of binary relationships between concepts: *is_a*, and *attribute*. The *is_a* relationship indicates that a concept is more specific (or conversely, more general) than another (e.g., the concept of *base price* is more specific than the concept of *price*); the *attribute* relationship simply indicates that a concept is an attribute of another (e.g., the *product* concept is the *priceOf* attribute of the *price* concept).

There are at least two approaches to ontology modeling. A widely practiced approach attempts to fully describe specializations so that there will be no ambiguity in the semantics of the most specific concepts (e.g., *basePrice_in_*1*'s_USD*) in an ontology. Such an ontology explicitly captures all contexts of the *price* concept in the example. These contexts are represented by the specialized concepts on the leaf level of a graphic representation of the ontology shown in Figure 1(a).

Another approach, developed in the Context Interchange (COIN) [Goh et al., 1999; Firat, 2003] project for semantic data integration purposes, only includes the most generic concepts; detailed definitions (i.e., specializations) of the general concepts are captured outside the ontology as localized context descriptions. To facilitate context description, the COIN ontology includes a special kind of attribute, called the *modifier*. Contexts are described by assigning values to modifiers. We call a COIN ontology a *lightweight ontology*.

These two different approaches are illustrated in Figure 1 using the price quote example.



(a) Fully-Specified Ontology

(b) Lightweight Ontology

Figure 1. Fully-specified v. lightweight ontologies.

As discussed in [Zhu, 2005; Madnick and Zhu, forthcoming] about the two ontology approaches, the lightweight ontology approach of COIN has several advantages, such as simplicity, maintainability, and flexibility to accommodate variations of the semantics of concepts. Below we compare the two approaches from the perspective of using ontology to represent and reason about contexts.

Both approaches provide structured descriptions for contexts. The fully-specified approach explicitly represents contexts in the ontology when the correspondence is established between the price of a vendor and a most

specialized price concept on the leaf level of the ontology. For example, context c_2 can be described by mapping the price data of vendor 2 to the left-most concept on the leaf level, which is *basePrice_in_1's_USD*.

In contrast, the lightweight ontology approach represents context outside the ontology. It does so by assigning values to modifiers present in the ontology. For example, context c_2 can be represented by a set of <*modifier*, *value*> pairs:

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

Although both approaches allow for structured context description and the structure can be exploited to reduce the number of lifting rules required, the fully-specified approach tends to invite implementations that require pair-wise lifting rules between contexts represented by the leaf concepts in the ontology. We call a lifting rule defined between different contexts a *composite lifting rule*.

With the lightweight ontology approach, simple lifting rules can be supplied for each modifier between different modifier values. For example, a lifting rule can be defined for *currency* modifier to convert values in different currencies using the exchange rate function shown earlier. We call a lifting rule defined for a modifier a *component lifting rule*. As we will see next, reasoning algorithms can be implemented to compose the composite lifting rules using component lifting rules. With this auto composition capability, we can significantly reduce the number of lifting rules needed.

Lifting Rule Composition

We use Figure 2 to illustrate the concept of lifting rule composition.

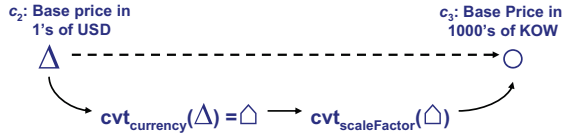


Figure 2. Composition of Lifting Rules.

In Figure 2, 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 lifting rules. The dashed straight arrow represents the application of composite lifting rule, such as rule (1) shown earlier using logic of context.

With the lightweight ontology approach, we can compose the composite lifting rule using predefined component lifting rules. As shown in Figure 2, we first apply the component lifting rule for *currency* modifier (represented by $cvt_{currency}$), then apply the component lifting rule for *scaleFactor* modifier (represented by $cvt_{scaleFactor}$).

The composition algorithm, shown in Figure 3, is quite simple. In COIN project, it is implemented using abductive constraint logic programming (ACLP) [Kakas et al., 2000].

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

Figure 3. Algorithm for lifting rule composition.

Benefit of Rule Composition

The primary benefit of the rule composition capability is the small number of component lifting rules required, thus increased scalability when many data sources and contexts are involved in data integration applications. A detailed analysis of scalability and other benefits can be found in [Zhu and Madnick, 2004; Zhu, 2005]; below we provide a brief discussion of the scalability benefit.

In the worst case, the number of component lifting rules required by the lightweight 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 for representing all contexts, m is the number of modifiers in the lightweight ontology.

Symbolic equation solver techniques have been developed to exploit the relationships between component lifting rules of a same modifier [Firat, 2003]. For example, consider a scenario where we have three definitions for price: (A) base price, (B) price with tax included, and (C) price with tax and shipping & handling included. This can be modeled by using a modifier that has three unique values for *price* concept in the ontology. With known equational relationships among the three price definitions, and two lifting rules:

(1) from base_price to base_price+tax (i.e., A to B) and
(2) from base_price+tax to base_price + tax + shipping & handling (i.e., B to C)

the symbolic equation solver can compute the other four conversions automatically (A to C and the three inverses). This technique further reduces the number of component lifting rules needed.

In many cases, the lifting rule for a modifier can be parameterized, i.e., the rule can be applied to convert for any given pair of modifier values. In this case, we only need to supply one component lifting rule for the modifier, regardless of the number of unique values that the modifier may have. With the exchange rate function given earlier, we only need one component lifting for currency modifier.

Let us use the online price comparison example to illustrate the benefit of the approach. With the given scenario, we can model the 30 unique contexts using the three modifiers in the lightweight ontology shown in Figure 1(b). Suppose the number of unique values of each modifier is as shown in Table 2. In the worst case, the lightweight ontology approach needs 102 (i.e., 90+6+6) component lifting rules. Since the lifting rules for *currency* and *scaleFactor* modifiers are parameterizable, the actual

number of component lifting rules needed is further reduced to 8, which is a significant improvement from the 870 composite lifting rules required when contexts are unstructured and lifting rules are specified pair-wise between contexts.

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

Related Work and Discussion

There has been a lot of work done to develop logic formalisms for contexts, characterize the computational complexities of the formalisms, and develop theories for reasoning about contexts. But not so much work has been done to characterize lifting rules. We are only aware of a recent effort by Guha and McCarthy [2003], who began looking at possible ways to reduce the number of lifting rules required. One way to reduce the number is by using default lifting rules, much like the use of frame default in dealing with the frame problem in AI. Another possible way is by writing more general lifting rules (e.g., instead of one rule per predicate, it may be possible to use one rule for all predicates that have certain characteristics). As a first step toward this direction, Guha and McCarthy [2003] provide a lifting rule categorization to reveal certain patterns of lifting rules. 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.

The unique characteristic of the COIN approach is in the use of lightweight ontology to provide a structure for context representation. The approach allows for explicit and structured context representation. In addition, it allows us to use a reasoning algorithm that exploits the context structure and provides the capability of lifting rule composition. The capability helps reduce the required number component lifting rules.

Ontology is used in [Kashyap and Sheth, 1996] to provide structured context representation. However, we are not certain if their ontology constitute a lightweight ontology. No assessment about the number of lifting rules required by their approach is given in their paper.

Ontology is also used in [Ram and Park, 2004], 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 Figure 1(a). 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 lifting rules needed is given in the paper.

A Context Ontology Language (CoOL) is introduced in [Strang et al., 2003], where an Aspect-Scale-Context (ASC) model is used to organize contexts and assist with specifications of lifting rules (which are called IntraOperations in the paper). Roughly speaking, the Aspect corresponds to the modifier, and the Scale corresponds to the specifications for a modifier in the COIN approach. The paper does not provide a characterization of the number of lifting rules needed by the ASC approach.

Conclusion

In this paper, we introduced structured contexts, where the structure is provided by a lightweight ontology. We also presented an algorithm that exploits the structure to reduce the number of required lifting rules. Since writing lift rules is a labor intensive and error-prone process, the requirement of a large number rules has been one of the major obstacles in large scale information integration projects. The lightweight ontology and structured context approach introduced in the paper provides an effective solution to the problem.

Since a lightweight ontology is simpler than a fully-specified one, we expect that it can be developed much more easily and used more widely. For future research, we would like to explore the applicability of this approach in other application domains, such as context-aware web services and peer-to-peer information sharing. Another promising area is to apply the context representation and reasoning techniques to Semantic Web applications. Initial work has been done [Tan et al., 2005] to represent COIN ontology and contexts using Semantic Web languages, such as OWL and RuleML. The preliminary results indicate that COIN lightweight ontology, structured context descriptions, and component lifting rules can be represented using Semantic Web languages. Future work will adapt the reasoning algorithm and evaluate its performance at large scales that are typical on the Semantic Web.

References

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.

Firat, A. (2003) "Information Integration using Contextual Knowledge and Ontology Merging", PhD Thesis, Sloan School of Management, MIT.

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.

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

Guha, R. and McCarthy, J. (2003) "Varieties of Contexts", *CONTEXT 2003***,** LNAI 2680, 164-177.

Guha, R., McCool, R. and Fikes, R. (2004) "Contexts for the Semantic Web", *ISWC'04 (LNCS 3298),* Japan, 32-46.

Kakas, A. C., Michael, A. and Mourlas, C. (2000) "ACLP: Abductive Constraint Logic Programming", *Journal of Logic Programming,* **44**(1-3)**,** 129-177.

Kashyap, V. and Sheth, A. P. (1996) "Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach", *VLDB Journal,* **5**(4)**,** 276-304.

Madnick, S. E. and Zhu, H. "Improving data quality through effective use of data semantics", *Data & Knowledge Engineering,* forthcoming.

McCarthy, J. (1987) "Generality in Artificial Intelligence", *Communications of the ACM,* **30**(12)**,** 1030-1035.

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.

Ram, S. and 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.

Strang, T., Linnhoff-Popien, C. and Frank, K. (2003) "CoOL: A Context Ontology Language to Enable Contextual Interoperability", In *DAIS 2003 (LNCS 2893)* (Eds, Stefani, J. B., Demeure, I. and Hagimont, D.), pp. 236-247.

Tan, P., Madnick, S. E. and Tan, K.-L. (2005) "Context Mediation in the Semantic Web: Handling OWL Ontology and Data Disparity Through Context Interchange", In *Semantic Web and Databases: Second International Workshop (SWDB 2004)*, Vol. LNCS 3372 (Eds, Bussler, C., Tannen, V. and Fundulaki, I.), pp. 140-154.

Zhu, H., Madnick, S. and Siegel, M. (2002) "Global Comparison Aggregation Services", *1st Workshop on E-Business,* Barcelona, Spain.

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.

Zhu, H. (2005) "Effective Information Integration and Reutilization: Solutions to Technological Deficiency and Legal Uncertainty", PhD, Engineering Systems Division, MIT.