Scalable Interoperability Through the Use of COIN Lightweight Ontology

Hongwei Zhu[1,2] and Stuart E. Madnick[1]

[1] Massachusetts Institute of Technology
Sloan School of Management
30 Wadsworth Street, E53-320, Cambridge, MA 02142, USA
{mrzhu,smadnick}@mit.edu
[2] Old Dominion University
College of Business and Public Administration
Constant 2079, Norfolk, VA 23529, USA
hzhu@odu.edu

Abstract. There are many different kinds of ontologies used for different purposes in modern computing. A continuum exists from lightweight ontologies to formal ontologies. In this paper we compare and contrast the lightweight ontology and the formal ontology approaches to data interoperability. Both approaches have strengths and weaknesses, but they both lack scalability because of the n^2 problem. We present an approach that combines their strengths and avoids their weaknesses. 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. It also provides a structure for context descriptions. The structure can be exploited to facilitate automatic composition of context mappings. This mechanism leads to a scalable solution to semantic interoperability among disparate data sources and contexts.

Keywords: lightweight ontology, formal ontology, context, mediation, scalability, semantic heterogeneity.

1 Introduction

Ontologies have been widely used in modern computing for purposes such as communication, computational inference, and knowledge organization and reuse [7]. For different purposes there are a variety of different ontologies, which 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 [21], 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. Both approaches can be used to support data interoperability among disparate sources.

M. Collard (Ed.): ODBIS 2005/2006, LNCS 4623, pp. 37–50, 2007.

Lightweight ontologies usually are taxonomies, which consist of a set of concepts (i.e., terms, or semantic types) and hierarchical relationships among the concepts. As an artifact, it is relatively easy to construct a lightweight ontology. However, such lightweight ontologies do not capture the detailed semantics of the concepts, which sometimes is documented in a data dictionary, and/or embedded in the data models and the data processing programs.

There are two different approaches to using lightweight ontologies for interoperability purposes. One approach is to develop a single lightweight ontology, in which case all parties need to agree on the exact meaning of the concepts. 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. However, reaching such agreements can be difficult. For example, a data standardization effort within the U.S. Department of Defense (DoD) took more than a decade only to standardize less than 2% of the data across all organizations of the DoD [18]. The alternative approach is to allow multiple lightweight ontologies to co-exist, in which case mappings among the ontologies need to be provided. Because the semantics is not formally captured in the ontologies, efforts are required to identify the semantic differences and then develop (often hand-code) the mappings to enable pair-wise interoperability. The number of pair-wise mappings is $n(n\text{-}1)$ (which is $O(n^2)$) if there are n different ontologies, thus the amount of effort required increases quickly as n becomes large. This is the so called n^2 problem of data interoperability. A survey [19] shows that approximately 70% of the costs of data interoperability projects are spent on identifying the semantic differences and developing code to reconcile them.

In contrast, the formal ontology approach uses axioms to explicitly represent semantics and has inference capabilities. This approach can also support interoperability either via a single ontology or via mappings of multiple ontologies. The key difference is that the semantics of the ontological concepts and the mappings are explicitly captured in a formal logic theory.

To summarize, both ontology approaches can be used to support data interoperability either via standardization or via mappings of multiple ontologies. The difficulty of reaching an agreement on a single data standard can be enormous so that in practice multiple standards (i.e., ontologies) co-exist even within a single organization. Thus, in practice ontology mappings are required to enable interoperability among data sources and systems. Both ontology approaches suffer from the n^2 problem. The key difference between the two ontology approaches is that lightweight ontologies do not capture the semantics in the ontologies, whereas formal ontologies explicitly capture semantics. As artifacts, lightweight ontologies are simple and easy to create, whereas formal ontologies are complex and difficult to create. But the semantics and the mappings of lightweight ontologies are often scattered in various data models and data processing programs, making maintenance extremely difficult. The semantics and mappings of formal ontologies are in the form of a logic theory, which is relatively easier to maintain. Both approaches have weaknesses that limit their effectiveness.

It is desirable to have an approach that combines the strengths and avoid the weaknesses of the two ontological approaches. In this paper, we present such an approach, which is developed in the COntext INterchange (COIN) project [3, 5, 25]

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, such as Cameleon [2] and others [1], and the increasing adoption of XML and 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 [22].

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. A *base price* refers to price with taxes and shipping & handling (S&H) excluded (e.g., price quotes from vendors 2 and 3).

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

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

simplicity, we will assume that users normally adopt a vendor context. Or we can assume that the only users are the vendors, each of whom wants to compare his prices with all of his world-wide competitors and wants the comparison done in his own 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 (e.g., prices in French francs and German deutschemarks became Euros), can be costly and error-prone.

2.2 COIN Lightweight Ontology

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

Like the traditional lightweight ontology, the COIN ontology includes a set of concepts, among which there can be a hierarchy represented with an *is_a* relationship. Besides, 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, the COIN lightweight ontology introduces *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.

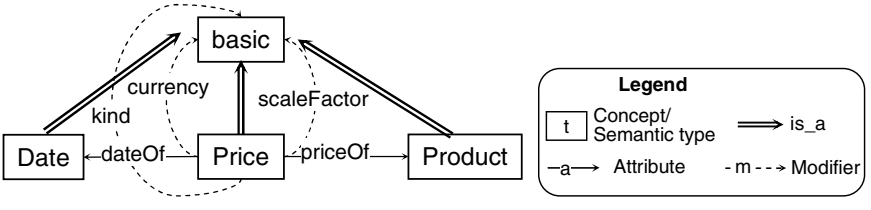

Fig. 1. COIN lightweight ontology for online price comparison example. It contains only high level concepts, the refined variants of which can be derived from the assignments of modifiers that belong to each high level concept.

In this ontology, we include a modifier-free root concept *basic*, 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. In simple cases, a specific value is assigned to a modifier in a context. In other cases, the assignment must be specified by a set of rules. In either case, a context is conceptually a set of assignments of all modifiers and can be described by a set of <*modifier*, *value*> pairs. For example, contexts c_2 and c_3 (refer to vendors 2 and 3 in Table 1) can be described as:

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

c_3 := { <*kind*, *basePrice*>, <*currency*, *krw*>, <*scaleFactor*, 1000> }

The language used in COIN for describing context (as well as context mappings and the lightweight ontology) is based on F-logic [12], an object-oriented logic. F-logic rules are converted to Datalog for reasoning purposes. In COIN, various "user-friendly" front-ends have been created so that developers do not directly need to use F-logic or Datalog. Below is example rule using the logic to assign a value to *currency* modifier in context c_3:

$$\forall X : price \; \exists Y : basic \vdash$$
$$X[currency(c_3) \rightarrow Y] \wedge Y[value(c_3) \rightarrow 'KRW'].$$

where variables (e.g., X, Y) are objects, the modifier and attributes of which are represented by methods (which are declared in square brackets). The method *value* is similar to the *value* predicate in context logic of [15]; it returns the ground value of the object in the context specified by the parameter (which is c_3 in the example).

2.3 Characteristics of COIN Lightweight Ontology

A COIN ontology, as shown in Fig. 1, includes only high level concepts (plus their relationships, such as the binary relationships of 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.

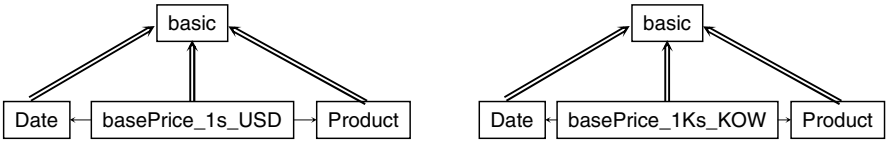

Fig. 2. Conceptual local ontologies for vendor 2 (left) and vendor 3 (right), derivable from COIN lightweight ontology shown in Fig. 1

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.

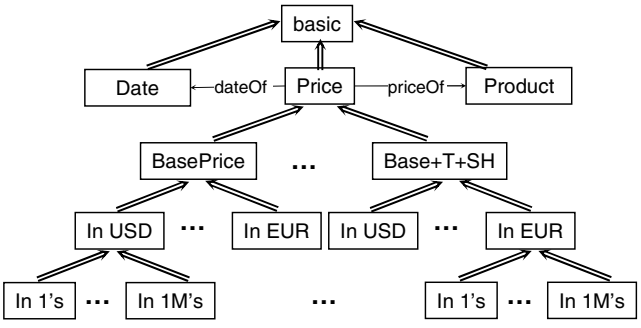

Fig. 3. An example fully-specified global ontology for the online price comparison example. Leaf nodes represents the concepts with specific semantics, e.g., the first leaf node on the left represent the concept of "price, not including taxes or shipping handling, in 1's of USD".

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 [13, 24], the COIN approach has several advantages over the formal ontology approach. 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 if a single ontology is used, or mappings need to be added to ensure its interoperability with other variants of the *price* concept. In contrast, the COIN approach can accommodate this new concept by adding new context descriptions without changing the ontology. As we will see later, the new mappings may not need to be added when they can be derived from existing mappings using a reasoning mechanism.

The COIN lightweight ontology approach also has advantages over the traditional lightweight ontology approach. Although, similar to the traditional approach, the

COIN ontology does not include detailed descriptions of semantics, it does provide a vocabulary and the structure for describing semantics using context descriptions. As we will see in the next section, the context reasoning mechanism exploits the structure to solve the n^2 problem.

3 Scalable Interoperability with COIN Lightweight Ontology

When data sources and data receivers are in different contexts, conversions (also called lifting rules or mappings) 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 hand-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 not specified as convoluted rules pair-wise between contexts. Instead, they 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:

$$olsen(CurFrom, CurTo, Day, Rate)$$

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*.

The component conversions in COIN are also specified using F-logic. Below is an example component conversion for currency modifier; it is parameterized with context C1 and C2 and can convert between any currencies. We use *olsen_* for the skolemized version of original *olsen* predicate.

$$\begin{aligned}
&\forall X : price \vdash \\
&\quad X[cvt(currency, C2) @ C1, u \rightarrow v] \leftarrow \\
&\qquad X[currency(C1) \rightarrow C_f] \wedge X[currency(C2) \rightarrow C_t] \wedge x[dataOf \rightarrow T] \wedge \\
&\qquad olsen_(A, B, R, D) \wedge C_f \overset{C2}{=} A \wedge C_t \overset{C2}{=} B \wedge T \overset{C2}{=} D \wedge R[value(C2) \rightarrow r] \wedge v = u * r.
\end{aligned}$$

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

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.

In Fig. 4, the triangle symbol on the left represents the price concept in context c_3, i.e., base price in 1000's of South Korean won (KRW); and the circle symbol on the right represents the price concept in context c_2, i.e., base price in 1's of USD. For data in context c_3 to be viewed in context c_2, they need to be appropriately converted by applying the appropriate composite conversion. The dashed straight arrow represents the application of the composite conversion that would have been implemented manually in other approaches. 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 $\text{cvt}_{\text{currency}}$), then apply the component conversion for *scaleFactor* modifier (represented by $\text{cvt}_{\text{scaleFactor}}$).

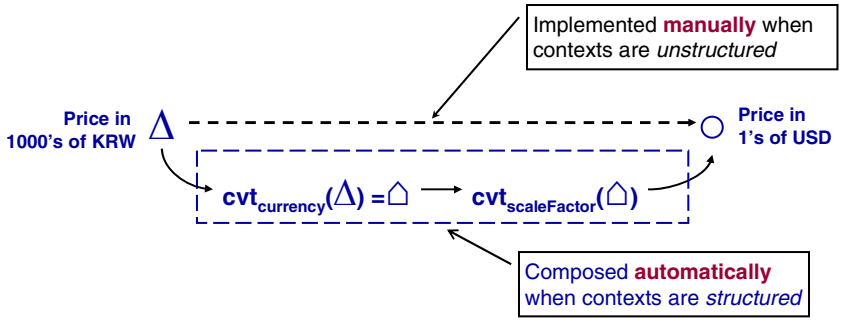

Fig. 4. Composite conversion composed using component conversions. Without composition, one would hand-code a direct conversion to convert the price in 1000's of KRW to the price in 1's of USD; this conversion illustrated by the straight dashed arrow. With COIN, this composite conversion can be derived from the component conversions for currency ($\text{cvt}_{\text{currency}}$) and scale factor ($\text{cvt}_{\text{scaleFactor}}$).

The composition algorithm, shown in Fig. 5, is quite simple. In COIN project, it is implemented in a query rewriting mediator using abductive constraint logic programming (ACLP) [10] and constraint handling rules (CHR) [4]. With the mediator, queries can be issued as if all data sources were in the requester's context (i.e., the target context). The mediator generates mediated queries that contain the composite conversions. Data is converted from source contexts to the requester's context when the mediated queries are executed.

A demonstration of the query mediator is shown in Fig. 6. The source used also includes a *Vendor* column, as shown in the sample schema near the middle of the figure. The source context corresponds to context c_3, and the requester context (c_c_usa2 in the figure) is equivalent to context c_2 in the online price comparison example discussed earlier. In the demonstration, the *QuoteDate* field can have different date formats, which we did not include in the ontology discussed earlier but can be accommodated by adding a *dateFormat* modifier to *Date* concept in the ontology in Fig. 1.

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

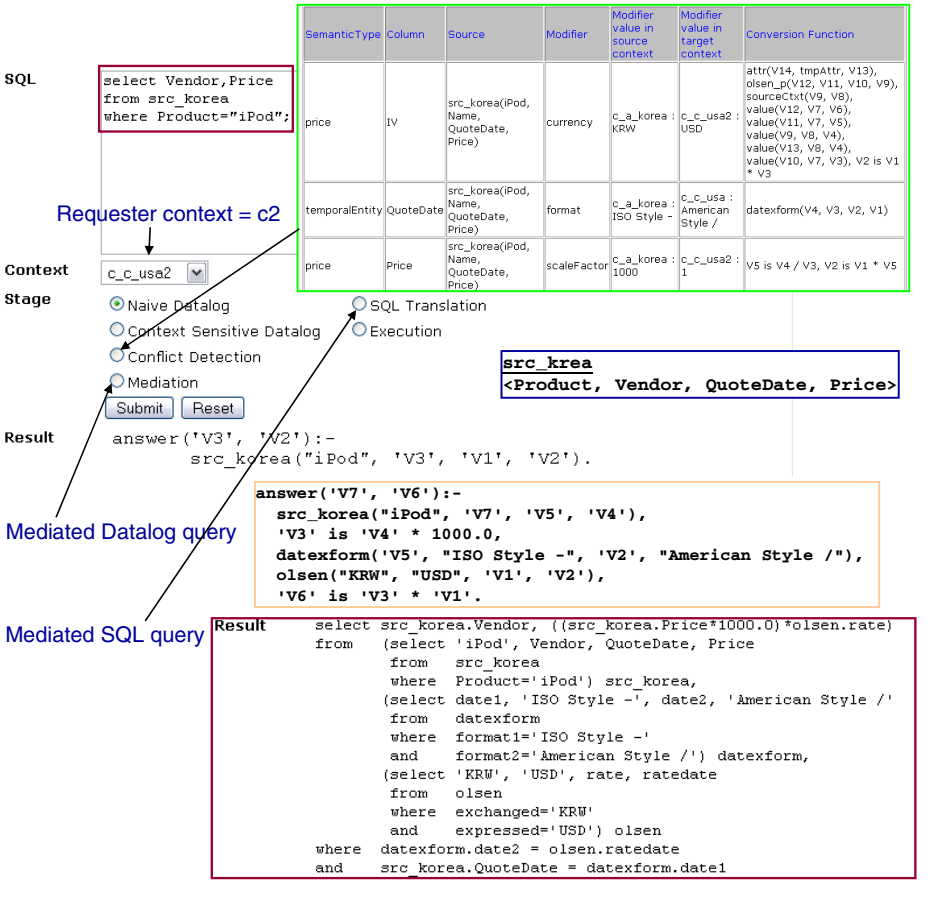

Fig. 6. A demonstration of conversion composition as query mediation

The requester SQL query, shown in the upper left of the figure, need not be aware of any context differences. Our demonstration system allows us to step through the various steps of mediation individually (e.g., converting the SQL to naïve Datalog query, etc.). The Conflict Detection step outputs a table that summarizes the concepts (called Semantic Types) whose modifiers have different values in the source and

requester contexts. A mediated Datalog query is generated using the algorithm shown in Fig. 5. As can be seen, the mediated query contains the necessary conversions to reconcile the context differences (namely currency and scale factor differences of *price* concept, which corresponds to the Price filed in the source table, and format difference of the *Date* concept, which corresponds to the *QuoteDate* field). The mediated Datalog query can be converted an SQL query, which is shown at the bottom in the figure.

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 [23, 24].

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.

While the formula appears to be n^2, it is fundamentally different from the approach that supplies *comprehensive conversions* between each pair of contexts. The supplied conversions in COIN are *component conversions,* which are much simpler than the comprehensive conversions that consider the differences of all data elements in all aspects between two contexts. Furthermore, as shown below, the number of component conversions required can be significantly smaller.

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. But 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.

The number of component conversions can be further reduced when equational relationships exist between contexts with different values of a modifier. Symbolic equation solver techniques have been developed to exploit such relationships [3]. For example, consider the three definitions for price: (A) base price, (B) price with tax

included, and (C) price with tax and shipping & handling included. With known equational relationships among the three price definitions, and two component conversions:

(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 conversions needed for a modifier from $n_i(n_i\text{-}1)$ to $(n_i\text{-}1)$.

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.

We use Fig. 7 to illustrate the intuition of the scalability result.

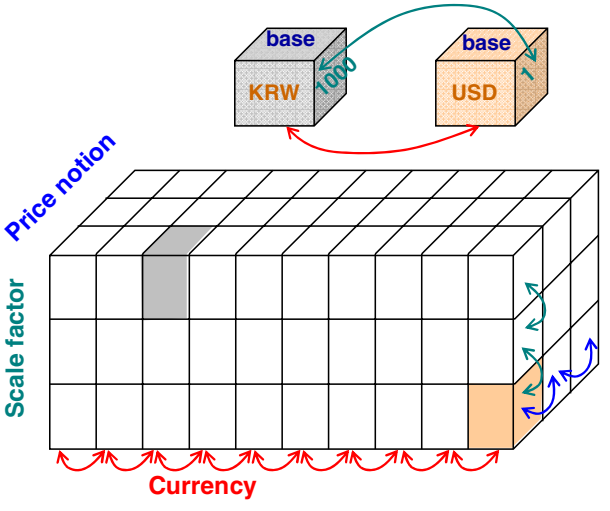

Fig. 7. Intuition of scalability of COIN approach. Component conversions are provided along the modifier axes. Composite conversions between any cubes in the space can be automatically composed.

The modifiers of each ontological concept span a context space within which the variants of the concept exist. Each modifier defines a dimension. In the figure, we show the space spanned by the three modifiers of *price* concept. The component conversions required by the COIN approach are defined along the axes of the modifiers. With the composition capability, the COIN approach can automatically generate all the conversions between units (e.g., the cubes in a three-dimensional space, as sown in Fig. 7) in the space using the component conversions along the dimensions. In contrast, the approaches that suffer from the n^2 problem require the conversions between any two units in the space to be supplied.

4 Related Work and Discussion

The most commonly cited definition for ontology is given in [6], where an ontology is a "formal explicit specification of a share conceptualization". But as discussed in [7, 20], 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 [8, 14].

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 [21]. 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 [15]. 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. Below is an example conversion rule to convert price in c_3 to price in c_2 by reconciling the currency and scale factor differences; the rule is a logic implementation of the conversion represented by the straight dashed line in Fig. 4:

$$c_0 : ist(c_2, quote(I, X, D)) \leftarrow ist(c_3, quote(I, P, D)), \quad olsen(krw, usd, D, R), X = P * R * 1000.$$

Suppose there n cubes in the contextual space shown in Fig. 7, the approach requires $n(n\text{-}1)$ conversion rules like the above one to enable full interoperability.

A recent effort tries to categorize lifting rules and attempts to use the patterns revealed to devise general lifting rules [9]. 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 [16], 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 [11] to provide structured context representation for purposes of data interoperability in a multi-database environment. 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 semantics of the concepts is described as context descriptions outside the ontology. It can be as a hybrid approach where are a lightweight ontology is annotated with a logic (i.e., F-logic) that can be in a formal ontology approach. 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. Another promising area is to apply the context represent-tation and reasoning techniques to Semantic Web applications. Initial work has been done [19] 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.

Acknowledgements. This work has been supported, in part, by The MITRE Corporation, the MIT-Malaysia University of Science and Technology (MUST) project, the Singapore-MIT Alliance (SMA), and Suruga Bank.

References

1. Chang, C.H., Kaye, M., Girgis, M.R., Shaalan, K.F.: A Survey of Web Information Extraction System. IEEE Transactions on Knowledge and Data Engineering 18(10), 1411–1428 (2006)
2. Firat, A., Madnick, S.E., Siegel, M.D.: The Cameleon Web Wrapper Engine. In: Workshop on Technologies for E-Services (TES'00), Cairo, Egypt (2000)
3. Firat, A.: Information Integration using Contextual Knowledge and Ontology Merging. In: PhD Thesis, Sloan School of Management. MIT, Cambridge, MA (2003)
4. Frühwirth, T.: Theory and Practice of Constraint Handling Rules. Journal of Logic Programming 37(1-3), 95–138 (1998)
5. Goh, C.H., Bressan, S., Madnick, S., Siegel, M.: Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. ACM Transitions on Information Systems 17(3), 270–293 (1999)

6. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–220 (1993)
7. Gruninger, M., Lee, J.: Ontology Applications and Design. Communications of the ACM 45(2), 39–41 (2002)
8. Guarino, N.: Formal Ontology and Information Systems. In: Guarino, N. (ed.) Proceedings of Formal Ontologies in Information Systems (FOIS '98), Trento, Italy, June 6-8, 1998, pp. 3–15. IOS Press, Amsterdam (1998)
9. Guha, R., McCarthy, J.: Varieties of Contexts. In: Blackburn, P., Ghidini, C., Turner, R.M., Giunchiglia, F. (eds.) CONTEXT 2003. LNCS, vol. 2680, pp. 164–177. Springer, Heidelberg (2003)
10. Kakas, A.C., Michael, A., Mourlas, C.: ACLP: Abductive Constraint Logic Programming. Journal of Logic Programming 44(1-3), 129–177 (2000)
11. Kashyap, V., Sheth, A.P.: Semantic and Schematic Similarities between Database Objects: A Context-Based Approach. VLDB Journal 5(4), 276–304 (1996)
12. Kiffer, M., Laussen, G., Wu, J.: Logic Foundations of Object-Oriented and Frame-based Languages. J. ACM 42(4), 741–843 (1995)
13. Madnick, S.E., Zhu, H.: Improving data quality through effective use of data semantics. Data & Knowledge Engineering 59(2), 460–475 (2006)
14. Mädsche, A.: Ontology Learning for the Semantic Web. Kluwer Academic Publishers, Boston, MA (2002)
15. McCarthy, J., Buvac, S.: Formalizing Context (Expanded Notes). In: Aliseda, A., van Glabbeek, R., Westerstahl, D. (eds.) Computing natural language, Sanford University (1997)
16. Ram, S., Park, J.: 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 (2004)
17. Rosenthal, A., Seligman, L., Renner, S.: From Semantic Integration to Semantics Management: Case Studies and a Way Forward. ACM SIGMOD Record 33(4), 44–50 (2004)
18. Seligman, L., Rosenthal, A., Lehner, P., Smith, A.: Data Integration: Where Does the Time Go? IEEE Bulletin of the Technical Committee on Data Engineering 25(3), 3–10 (2002)
19. Tan, P., Madnick, S.E., Tan, K.-L.: Context Mediation in the Semantic Web: Handling OWL Ontology and Data Disparity Through Context Interchange. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372, pp. 140–154. Springer, Heidelberg (2005)
20. Uschfold, M., Gruninger, M.: Ontologies and Semantics for Seamless Connectivity. ACM SIGMOD Record 33(4), 58–64 (2004)
21. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-Based Integration of Information - A Survey of Existing Approaches. In: IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, pp. 108–117 (2001)
22. Zhu, H., Madnick, S., Siegel, M.: Global Comparison Aggregation Services. In: 1st Workshop on E-Business, Barcelona, Spain (2002)
23. Zhu, H., Madnick, S.E: Context Interchange as a Scalable Solution to Interoperating Amongst Heterogeneous Dynamic Services. In: 3rd Workshop on eBusiness (WEB), Washington, D.C., pp. 150–161 (2004)
24. Zhu, H.: Effective Information Integration and Reutilization: Solutions to Technological Deficiency and Legal Uncertainty. In: Ph.D. Thesis. MIT, Cambridge, MA (2005)