Context Interchange: New Features and Formalisms for the Intelligent Integration of Information

CHENG HIAN GOH
National University of Singapore
and
STÉPHANE BRESSAN, STUART MADNICK, and MICHAEL SIEGEL
Massachusetts Institute of Technology

The *Context Interchange strategy* presents a novel perspective for mediated data access in which semantic conflicts among heterogeneous systems are not identified a priori, but are detected and reconciled by a *context mediator* through comparison of *contexts axioms* corresponding to the systems engaged in data exchange. In this article, we show that queries formulated on shared views, export schema, and shared "ontologies" can be mediated in the same way using the *Context Interchange framework*. The proposed framework provides a logic-based object-oriented formalism for representing and reasoning about data semantics in disparate systems, and has been validated in a prototype implementation providing mediated data access to both traditional and web-based information sources.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Query processing*; H.2.5 [**Database Management**]: Heterogeneous Databases—*Data translation*

General Terms: Design

Additional Key Words and Phrases: Abductive reasoning, information integration, mediators, semantic heterogeneity, semantic interoperability

We dedicate this work to the memory of Cheng Hian Goh (1965–1999).

This work is supported in part by ARPA and USAF/Rome Laboratory under contract F30602-93-C-0160, the International Financial Services Research Center (IFSRC), the PROductivity From Information Technology (PROFIT) project at MIT, and TASC, Inc., as well as NUS RP970628.

Authors' addresses: S. Bressan, S. Madnick, and M. Siegel, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139; email: sbressan@mit.edu; smadnick@mit.edu; msiegel@mit.edu.

1. INTRODUCTION

The number of online information sources and receivers has grown at an unprecedented rate in the last few years, contributed in large part by the exponential growth of the Internet as well as advances in telecommunications technologies. Nonetheless, this increased *physical connectivity* (the ability to exchange bits and bytes) does not necessarily lead to *logical connectivity* (the ability to exchange information meaningfully). This problem is sometimes referred to as the need for *semantic interoperability* [Sheth and Larson 1990] among *autonomous* and *heterogeneous* systems.

The *Context Interchange strategy* [Siegel and Madnick 1991; Sciore et al. 1994] is a mediator-based approach [Wiederhold 1992] for achieving semantic interoperability among heterogeneous sources *and* receivers, constructed on the following tenets:

—the *detection* and *reconciliation* of semantic conflicts are system services which are provided by a *Context Mediator*, and should be transparent to a user; and

—the provision of such a mediation service requires only that the user furnish a logical (declarative) specification of *how data are interpreted* in sources and receivers, and *how conflicts, when detected, should be resolved*, but *not what conflicts exists a priori* between any two systems.

This approach toward semantic interoperability is unlike most traditional integration strategies which either require users to engage in the detection and reconciliation of conflicts (in the case of *loosely coupled systems*, e.g., MRDSM [Litwin and Abdellatif 1987], VIP-MDBMS [Kuhn and Ludwig 1988]), or insist that conflicts should be identified and reconciled, a priori, by some system administrator, in one or more shared schemas (as in *tightly coupled systems*, e.g., Multibase [Landers and Rosenberg 1982] and Mermaid [Templeton et al. 1987]). In addition, the proposed framework plays a complementary role to an emerging class of integration strategies [Levy et al. 1995b; Ullman 1997] where queries are formulated on an "ontology" without specifying a priori what information sources are relevant for the query. Although the use of a logical formalism for information integration is not new (see, for example, Catarci and Lenzerini [1993] where interschema dependencies are represented using *description logics*), our integration approach is different because we have chosen to focus on the semantics of individual data items as opposed to conflicts at the schematic level.

With the above observations in mind, our goal in this article is (1) to illustrate various novel features of the Context Interchange mediation strategy and (2) to describe how the underlying representation and reasoning can be accomplished within a formal *logical* framework. Even though this work originated from a long-standing research program, the features and formalisms presented in this article are new with respect to our previous works. Our proposal is also capable of supporting "multidatabase" queries, queries on "shared views," as well as queries on shared "ontologies," while allowing semantic descriptions of disparate sources to remain loosely coupled to one another. The feasibility of this work has also been validated in a prototype system which provides access to both traditional data sources (e.g., Oracle data systems) as well as semistructured information sources (e.g., Web sites).

The rest of this article is organized as follows. Following this introduction, we present a motivational example which is used to highlight the Context Interchange strategy. Section 3 describes the Context Interchange framework by introducing both the representational formalism and the logical inferences underlying query mediation. Section 4 compares the Context Interchange strategy with other integration approaches which have been reported in the literature. The last section presents a summary of our contributions and describes some ongoing thrusts.

Due to space constraints, we have aimed at providing the intuition by grounding the discussion in examples where possible; a substantively longer version of the article, presenting more of the technical details, is available as a working paper [Goh et al. 1996]. A report on the Prototype can also be found in Bressan et al. [1997a]. An in-depth discussion of the context mediation procedure can be found in a separate report [Bressan et al. 1997b].

As one might easily gather from examining the literature, research in information integration is making progress in leaps and bounds. A detailed discussion on the full variety of integration approaches and their accomplishments is beyond the scope of this article, and we gladly recommend Hull [1997] for a comprehensive survey.

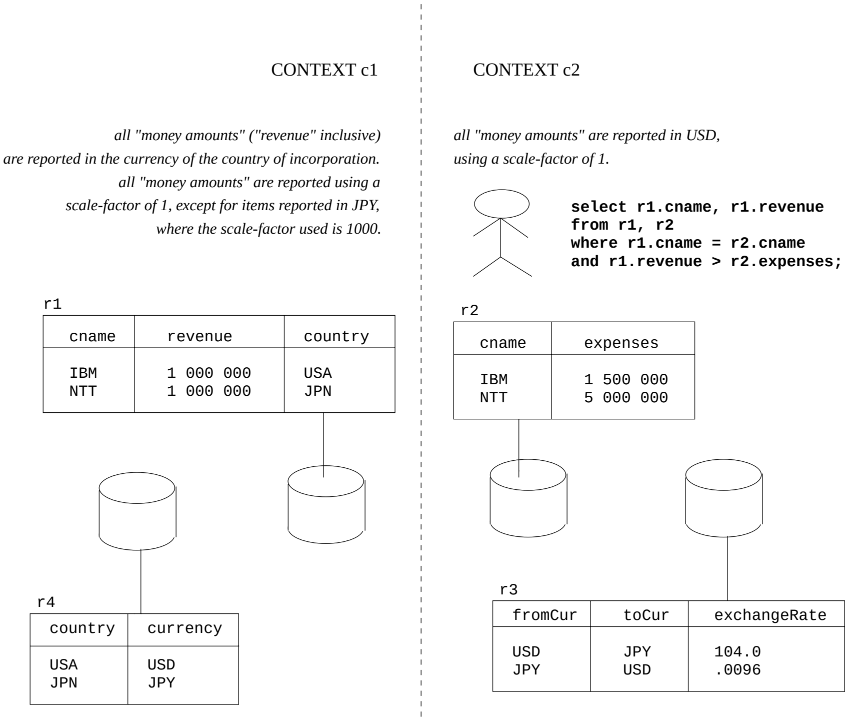
2. CONTEXT INTERCHANGE BY EXAMPLE

Consider the scenario shown in Figure 1, deliberately kept simple for didactical reasons. Data on "revenue" and "expenses" (respectively) for some collection of companies are available in two autonomously administered data sources, each comprised of a single relation denoted by r1 and r2 respectively. Suppose a user is interested in knowing which companies have been "profitable" and their respective revenue: this query can be formulated directly on the (export) schemas of the two sources as follows:

```
Q1: SELECT r1.cname, r1.revenue FROM r1, r2
    WHERE r1.cname = r2.cname AND r1.revenue > r2.expenses;
```

(We assume, without loss of generality, that relation names are unique across all data sources. This can always be accomplished via some renaming scheme: say, by prefixing the relation name with the name of the data source (e.g., db1#r1).) In the absence of any mediation, this query will return the empty answer if it is executed over the extensional data set shown in Figure 1.

The above query, however, does not take into account the fact that both sources and receivers may have different *contexts*, i.e., they may embody different assumptions on how information present should be interpreted. To simplify the ensuing discussion, we assume that the data reported in the two sources differ only in the currencies and scale-factors of "money



all "money amounts" ("revenue" inclusive) are reported in the currency of the country of incorporation. all "money amounts" are reported using a scale-factor of 1, except for items reported in JPY, where the scale-factor used is 1000.

all "money amounts" are reported in USD, using a scale-factor of 1.

```
select r1.cname, r1.revenue
from r1, r2
where r1.cname = r2.cname
and r1.revenue > r2.expenses;
```

r1

cname	revenue	country
IBM	1 000 000	USA
NTT	1 000 000	JPN

r2

cname	expenses
IBM	1 500 000
NTT	5 000 000

r4

country	currency
USA	USD
JPN	JPY

r3

fromCur	toCur	exchangeRate
USD	JPY	104.0
JPY	USD	.0096

Fig. 1. Example scenario.

amounts." Specifically, in Source 1, all "money amounts" are reported using a scale-factor of 1 and the currency of the country in which the company is "incorporated"; the only exception is when they are reported in Japanese Yen (JPY); in which case the scale-factor is 1000. Source 2, on the other hand, reports all "money amounts" in USD using a scale-factor of 1. In the light of these remarks, the (empty) answer returned by executing Q1 is clearly not a "correct" answer, since the revenue of NTT (9,600,000 USD = 1,000,000 × 1,000 × 0.0096) is numerically larger than the expenses (5,000,000) reported in r2. Notice that the derivation of this answer requires access to other sources (r3 and r4) not explicitly named in the user query.

In a Context Interchange system, the semantics of data (of those present in a source, or of those expected by a receiver) can be explicitly represented in the form of a *context theory* and a set of *elevation axioms* with reference to a *domain model* (more about these later). As shown in Figure 2, queries submitted to the system are intercepted by a *Context Mediator*, which rewrites the user query to a *mediated query*. The *Optimizer* transforms this to an optimized query plan, which takes into account a variety of cost information. The optimized query plan is executed by an *Executioner* which dispatches subqueries to individual systems, collates the results, undertakes conversions which may be necessary when data are exchanged between two systems, and returns the answers to the receiver. In the

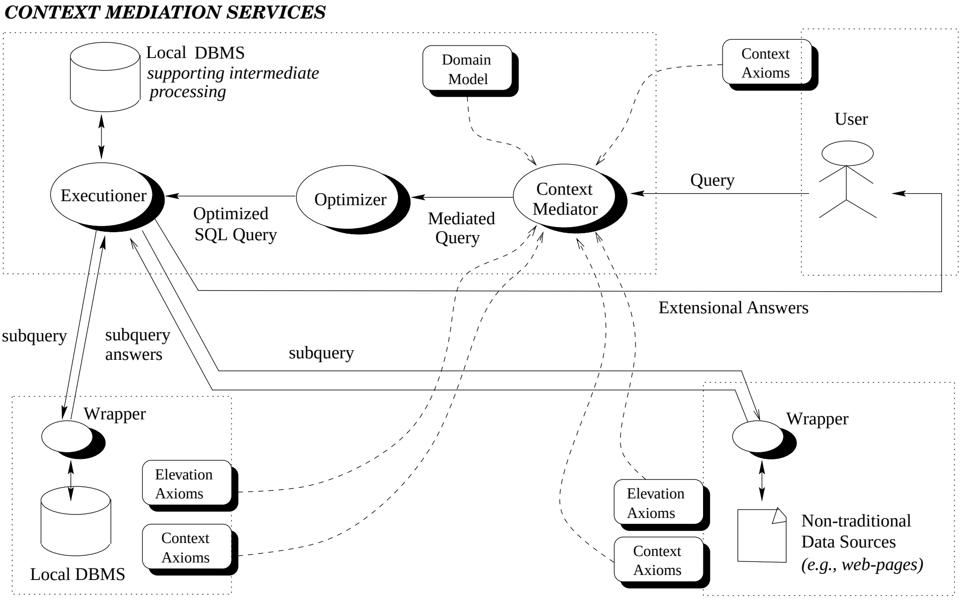


Fig. 2. Architecture of a Context Interchange system.

remainder of this section, we describe three different paradigms for supporting data access using this architecture.

2.1 Mediation of “Multidatabase” Queries

The query Q1 shown above is in fact similar to “multidatabase” MDSL queries described in Litwin and Abdellatif [1987] whereby the export schemas of individual data sources are explicitly referenced. Nonetheless, unlike the approach advocated in Litwin and Abdellatif [1987], users remain insulated from underlying semantic heterogeneity, i.e., they are not required to undertake the detection or reconciliation of potential conflicts between any two systems. In the Context Interchange system, this function is assumed by the Context Mediator: for instance, the query Q1 is transformed to the mediated query MQ1:

```
MQ1: SELECT r1.cname, r1.revenue FROM r1, r2, r4
     WHERE r1.country = r4.country
     AND r4.currency = 'USD'
     AND r1.cname = r2.cname
     AND r1.revenue > r2.expenses;
     UNION
     SELECT r1.cname, r1.revenue * 1000 * r3.rate
     FROM r1, r2, r3, r4
     WHERE r1.country = r4.country
     AND r4.currency = 'JPY'
     AND r1.cname = r2.cname
     AND r3.fromCur = 'JPY'
     AND r3.toCur = 'USD'
     AND r1.revenue * 1000 * r3.rate > r2.expenses
```

```
        UNION
        SELECT r1.cname, r1.revenue * r3.rate
        FROM r1, r2, r3, r4
        WHERE r1.country = r4.country
        AND r4.currency 〈〉 'USD'
        AND r4.currency 〈〉 'JPY'
        AND r3.fromCur = r4.currency
        AND r3.toCur = 'USD'
        AND r1.cname = r2.cname
        AND r1.revenue * r3.rate > r2.expenses;
```

This mediated query considers all potential conflicts between relations r1 and r2 when comparing values of "revenue" and "expenses" as reported in the two different contexts. Moreover, the answers returned may be further transformed so that they conform to the context of the receiver. Thus in our example, the revenue of NTT will be reported as 9 600 000 as opposed to 1 000 000. More specifically, the three-part query shown above can be understood as follows. The first subquery takes care of tuples for which revenue is reported in USD using scale-factor 1; in this case, there is no conflict. The second subquery handles tuples for which revenue is reported in JPY, implying a scale-factor of 1000. Finally, the last subquery considers the case where the currency is neither JPY nor USD, in which case only currency conversion is needed. Conversion among different currencies is aided by the ancillary data sources r3 (which provides currency conversion rates) and r4 (which identifies the currency in use corresponding to a given country). The mediated query MQ1, when executed, returns the "correct" answer consisting only of the tuple 〈'NTT', 9 600 000〉.

2.2 Mediation of Queries on "Shared Views"

Although "multidatabase" queries may provide users with greater flexibility in formulating a query, they also require users to know what data are present *in which data sources* and be sufficiently familiar with the attributes in different schemas (so as to construct a query). An alternative advocated in the literature is to allow *views* to be defined on the source schemas and have users formulate queries based on the view instead. For example, we might define a view on relations r1 and r2, given by

```
    CREATE VIEW v1 (cname, profit) AS
    SELECT r1.cname, r1.revenue - r2.expenses
    FROM r1, r2
    WHERE r1.cname = r2.cname;
```

in which case, query Q1 can be equivalently formulated on the view v1 as

```
    VQ1: SELECT cname, profit FROM v1
         WHERE profit > 0;
```

While this view approach achieves essentially the same functionalities as tightly coupled systems, *notice that view definitions in our case are no longer concerned with semantic heterogeneity and make no attempts at identifying or resolving conflicts, since query mediation can be undertaken* *by the Context Mediator as before.* Specifically, queries formulated on the shared view can be easily rewritten to queries referencing sources directly, which allows it to undergo further transformation by the Context Mediator as before.

2.3 Mediation of Queries on Shared "Ontologies"

Yet another approach for achieving read-only integration is to define a shared domain model (often called an *ontology*), which serves as a global schema identifying all information relevant to a particular application domain. However, unlike the traditional tight-coupling approach, data held in the source databases is expressed as views over this global schema [Levy et al. 1995b; Ullman 1997]. This means that queries formulated on the ontology must be transformed to "equivalent" queries on actual data sources.

It is important to note that current work in this direction has been largely focused on designing algorithms for realizing query rewriting with the goal of identifying the relevant information sources that must be accessed to answer a query (see, for example, Levy et al. [1995a] and Ullman [1997]). In all instances that we know of, it is assumed that no semantic conflicts whatsoever exist among the disparate data sources. It should be clear that the work reported here complements rather than competes with this "new wave" integration strategy.

3. THE CONTEXT INTERCHANGE FRAMEWORK

McCarthy [1987] points out that statements about the world are never always true or false: the truth or falsity of a statement can only be understood with reference to a given *context*. This is formalized using assertions of the form

$$\bar{c}: \quad ist(c, \sigma)$$

which suggests that the statement σ is true in ("*ist*") the context c, this statement itself being asserted in an *outer context* $\bar{c}$.

McCarthy's notion of "contexts" provides a useful framework for modeling statements in heterogeneous databases which are seemingly in conflict with one another: specifically, factual statements present in a data source are not "universal" facts about the world, but are true relative to the context associated with the source but not necessarily so in a different context. Thus, if we assign the labels c1 and c2 to contexts associated with sources 1 and 2 in Figure 1, we may now write

$\bar{c}$: *ist*(c1,r1("NTT", 1 000 000, "JPN")).
$\bar{c}$: *ist*(c2, r2("NTT", 5 000 000)).

where $\bar{c}$ refers to the ubiquitous context associated with the integration exercise. For simplicity, we will omit $\bar{c}$ in the subsequent discussion, since the context for performing this integration remains invariant.

The Context Interchange framework constitutes a formal, logical specification of the components of a Context Interchange system. This comprises three components:

—The *domain model* is a collection of "rich" types, called *semantic types*, which defines the application domain (e.g., medical diagnosis, financial analysis) corresponding to the data sources which are to be integrated.

—The *elevation axioms* corresponding to each source identify the correspondences between attributes in the source and semantic types in the domain model. In addition, it codifies the integrity constraints pertaining to the source; although the integrity constraints are not needed for identifying sound transformations on user queries, they are useful for simplifying the underlying representation and for producing queries which are more optimal.

—The *context axioms*, corresponding to named contexts associated with different sources or receivers, define alternative interpretations of the *semantic objects* in different contexts. Every source or receiver is associated with exactly one context (though not necessarily unique, since different sources or receivers may share the same context). We refer to the collection of context axioms corresponding to a given context c as the *context theory* for c.

The assignment of sources to contexts is modeled explicitly as part of the Context Interchange framework via a source-to-context mapping μ. Thus, $\mu(s) = c$ indicates that the context of source s is given by c. The functional form is chosen over the usual predicate-form (i.e., $\mu(s, c)$) to highlight the fact that every source can only be assigned exactly one context. By abusing the notation slightly, we sometimes write $\mu(r) = c$ if r is a relation in source s. As we shall see later on, the context of *receivers* is modeled explicitly as part of a query.

In the remaining subsections, we describe each of the above components in turn. This is followed by a description of the logical inferences—called abduction—for realizing query mediation. The Context Interchange framework is constructed on a deductive and object-oriented data model (and language) of the family of F(rame) logic [Kifer et al. 1995], which combines both features of object-oriented and deductive data models. The syntax and semantics of this language will be introduced informally throughout the discussion, and we sometimes alternate between an F-logic and a predicate calculus syntax to make the presentation more intuitive. This is no cause for alarm, since it has been repeatedly shown that one syntactic form is equivalent to the other (see, for instance, Abiteboul et al. [1993]). Notwithstanding this, the adoption of an "object-oriented" syntax provides us with greater flexibility in representing and reusing data semantics captured in different contexts. This is instrumental in defining an integration infrastructure that is *scalable*, *extensible*, and *accessible* [Goh et al. 1994]. This

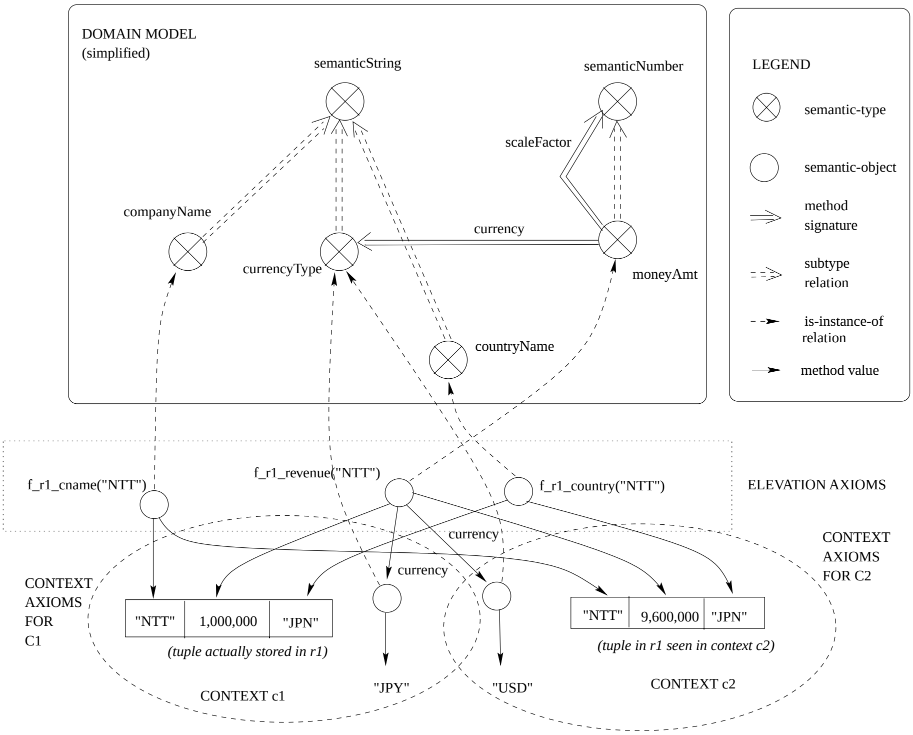


Fig. 3. A graphical illustration of the different components of a Context Interchange framework.

observation will be revisited in Section 4 where we compare our approach to the integration strategy adopted in Carnot [Collet et al. 1991].

3.1 The Domain Model

We distinguish between two kinds of data objects in the COIN data model: *primitive objects*, which are instances of *primitive types*, and *semantic objects* which are instances of *semantic types*. Primitive types correspond to data types (e.g., strings, integers, and reals) which are native to sources and receivers. Semantic types, on the other hand, are complex types introduced to support the underlying integration strategy. Specifically, semantic objects may have properties, called *modifiers*, which serve as annotations that make explicit the semantics of data in different contexts. Every object is identifiable using a unique *object-id* (OID) and has a value (not necessarily distinct). In the case of primitive objects, we do not distinguish between the OID and its value. Semantic objects, on the other hand, may have distinct values in different context. Examples of these will be presented shortly.

A *domain model* is a collection of primitive types and semantic types which provides a common type system for information exchange between disparate systems. A (simplified) domain model corresponding to our motivational example in Section 2 can be seen in Figure 3. We use a different symbol for types and object instances, and different arrow types to illustrate the disparate relationships between these. For example, double-shaft arrows indicate "signatures" and identify what modifiers are defined for each type, as well as the type of the object which can be assigned to the (modifier) slot. The notation used should be self-explanatory from the accompanying legend.

As in other "object-oriented" formalisms, types may be related in an abstraction hierarchy where properties of a type are inherited. This inheritance can be *structural* or *behavioral*: the first refers to the inheritance of the type structure, and the second, that of values assigned to instances of those types. For example, `semanticNumber`, `moneyAmt`, and `semanticString` are all semantic types. Moreover, `moneyAmt` is a subtype of `semanticNumber` and has modifiers `currency` and `scaleFactor`. If we were to introduce a subtype of `moneyAmt`, say `stockPrice`, into this domain model, then `stockPrice` will inherit the modifiers `currency` and `scaleFactor` from `moneyAmt` by structural inheritance. If we had indicated that all (object) instances of `moneyAmt` will be reported using a `scaleFactor` of `1`, this would be true of all instances of `stockPrice` as well by virtue of behavioral inheritance (unless this value assignment is overridden).

The object labeled `f_r1_revenue("NTT")` is an example of a semantic object, which is an instance of the semantic type `moneyAmt` (indicated by the dashed arrow linking the two). The token `f_r1_revenue("NTT")` is the unique OID and is invariant under all circumstances. Semantic objects are "virtual" objects, since they are never physically materialized for query processing, but exist merely for query mediation. As we will demonstrate in the next section, this object is defined by applying a Skolem function on the key-value of a tuple in the source. It is important to point out that a semantic object may have different values in different "contexts." Suppose we introduce two contexts labeled as `c1` and `c2` which we associate with sources and receiver as indicated in Figure 3. We may write

```
f_r1_revenue("NTT")[value(c1) → 1000000].
f_r1_revenue("NTT")[value(c2) → 9600000].
```

The above statements illustrate statements written in the COIN language (COINL), which mirrors closely that of F-logic [Kifer et al. 1995]. The token `value(c1)` is a *parameterized method* and is said to return the value `1000000` when invoked on the object `f_r1_revenue("NTT")`. The same statements could have been written using a predicate calculus notation:

```
ist(c1, value(f_r1_revenue("NTT"),1000000)).
ist(c2, value(f_r1_revenue("NTT"),9600000)).
```

The choice of an object logic however allows certain features (e.g., inheritance and overridding) to be represented more conveniently.

3.2 Elevation Axioms

Elevation axioms provide the means for mapping "values" present in sources to "objects" which are meaningful with respect to a domain model.

This is accomplished by identifying the semantic type corresponding to each attribute in the export schema, and in allowing semantic objects to be instantiated from values present in the source. In the graphical interface which is planned for the existing prototype, this is simply accomplished by scrolling through the domain model and "clicking" on the semantic type that corresponds to a given attribute that is to be exported by the current source.

Internally, this mapping of attributes to semantic types is formally represented in two different sets of assertions. We present below the abstract syntax of the language, which emphasizes the "logical" character of our representation. A concrete syntax, a lá OQL, is being developed for end-users and applications programmers to make the representation more accessible.

The first group of axioms introduces a semantic object corresponding to each attribute of a tuple in the source. For example, the statement

$$\forall x\ \forall y\ \forall z\ \exists u \text{ s.t. } u : \texttt{moneyAmt} \leftarrow \texttt{r1}(x, y, z)$$

asserts that there exists some semantic object u of type moneyAmt corresponding to each tuple in relation r1. This statement can be rewritten into the *Horn clause* [Lloyd 1987], where all variables are assumed to be universally quantified:

$$\texttt{f\_r1\_revenue}(x, y, z) : \texttt{moneyAmt} \leftarrow \texttt{r1}(x, y, z).$$

The existentially quantified variable u is replaced by the *Skolem object* [Lloyd 1987] f_r1_revenue(x, y, z). Notice that the *Skolem function* (f_r1_revenue) is chosen such that it is guaranteed to be unique. In this example, it turns out that the functional dependency cname → {revenue,country} holds on r1: this allows us to replace f_r1_revenue(x, y, z) by f_r1_revenue(x) without any loss of generality. This follows trivially from the fact that whenever we have f_r1_revenue(x, y, z) and f_r1_revenue(x, y', z'), it must be that $y = y'$ and $z = z'$ (by virtue of the functional dependency).

The second assertion is needed to provide the assignment of values to the (Skolem) semantic objects created before. We may thus write

$$\texttt{f\_r1\_revenue}(x)[\texttt{value}(c) \rightarrow y] \leftarrow \texttt{r1}(x, y, z),\ \mu(\texttt{r1}, c).$$

Consider, for instance, the semantic object f_r1_revenue("NTT") shown in Figure 3. This object is instantiated via the application of the first assertion. The second assertion allows us to assign the value 1000000 to this object in context c1, which is the context associated with relation r1. The value of this semantic object may however be different in another context, as in the case of c2. The transformation on the values of semantic objects between different contexts is addressed in the next subsection.

3.3 Context Axioms

Context axioms associated with a source or receiver provide for the articulation of the data semantics which are often implicit in the given context.

These axioms come in two parts. The first group of axioms defines the semantics of data at the source or receiver in terms of values assigned to modifiers corresponding to semantic objects. The second group of axioms complements this declarative specification by introducing the "methods" (i.e., *conversion functions*) that define how values of a given semantic object are transformed between different contexts.

Axioms of the first type takes the form of a first-order statement which make assignments to modifiers. Returning to our earlier example, the fact that all moneyAmt in context c2 are reported in US Dollars using a scale-factor of 1 is made explicit in the following axioms:

$$\begin{aligned}
&x : \texttt{moneyAmt},\ y : \texttt{semanticNumber} \vdash y[\texttt{value(c2)} \rightarrow \texttt{1}] \\
&\qquad \leftarrow x\,[\texttt{scaleFactor(c2)} \rightarrow y]. \\
&x : \texttt{moneyAmt},\ y : \texttt{currencyType} \vdash y[\texttt{value(c2)} \rightarrow \text{“}\texttt{USD}\text{”}] \\
&\qquad \leftarrow x\,[\texttt{currency(c2)} \rightarrow y].
\end{aligned}$$

In the above statements, the part preceding the symbol "⊢" constitutes the *predeclaration* identifying the object type(s) (class) for which the axiom is applicable. This is similar to the approach taken in Gulog [Dobbie and Topor 1995]. By making explicit the types to which axioms are attached, we are able to simulate nonmonotonic inheritance through the use of negation, as in Abiteboul et al. [1993].

The semantics of data embedded in a given context may be arbitrarily complex. In the case of context c1, the currency of moneyAmt is determined by the country-of-incorporation of the company which is being reported on. This in turn determines the scale-factor of the amount reported; specifically, money amounts reported using "JPY" uses a scale-factor of 1000, whereas all others are reported in 1's. The corresponding axioms for these are shown below:

$$\begin{aligned}
&x : \texttt{moneyAmt},\ y : \texttt{currencyType} \vdash y[\texttt{value(c1)} \rightarrow v] \leftarrow \\
&\qquad\qquad x[\texttt{currency(c1)} \rightarrow y],\ x = \texttt{f\_r1\_revenue}(u), \\
&\qquad\qquad \texttt{r1}(u, \_, w),\ \texttt{r4}(w, v). \\
&x : \texttt{moneyAmt},\ y : \texttt{semanticNumber} \vdash y\ [\texttt{value(c1)} \rightarrow \texttt{1000}] \leftarrow \\
&\qquad\qquad x[\texttt{scaleFactor(c1)} \rightarrow y;\ \texttt{currency(c1)} \rightarrow z], \\
&\qquad\qquad z[\texttt{value(c1)} \rightarrow v],\ v = \text{“}\texttt{JPY}\text{”}. \\
&x : \texttt{moneyAmt},\ y : \texttt{semanticNumber} \vdash y\ [\texttt{value(c1)} \rightarrow \texttt{1}] \leftarrow \\
&\qquad\qquad x[\texttt{scaleFactor(c1)} \rightarrow y;\ \texttt{currency(c1)} \rightarrow z], \\
&\qquad\qquad z[\texttt{value(c1)} \rightarrow v],\ v \neq \text{“}\texttt{JPY}\text{”}.
\end{aligned}$$

Following Prolog's convention, the token "_" is used to denote an "anonymous" variable. In the first axiom above, r4 is assumed to be in the same context as r1 and is assumed to constitute an ancillary data source for defining part of the context (in this case, the currency used in reporting moneyAmt). Bear in mind also that variables are local to a clause; thus, variables having the same name in different clauses have no relation to one another.

The preceding declarations are not yet sufficient for resolving conflicting interpretations of data present in disparate contexts, since we have yet to define how values of a (semantic) object in one context are to be reported in a different context with different assumptions (i.e., modifier values). This is accomplished in the Context Interchange framework via the introduction of *conversion functions* (methods) which form part of the context axioms. The conversion functions define, for each modifier, how representations of an object of a given type may be transformed to comply with assumptions in the local context. For example, scale-factor conversions in context c1 can be defined by multiplying a given value with the appropriate ratio as shown below:

```
x :moneyAmt ⊢
    x[cvt(scaleFactor,c1)@c,u → v] ←
          x[scaleFactor(c1) →  _[value(c1) → f]],
          x[scaleFactor(c) →  _[value(c1) → g]],
          v = u * g/f.
```

In the "antecedent" of the statement above, the first literal returns the scale-factor of x in context c1. In contrast, the second literal returns the scale-factor of x in some parameterized context c. c and c1 are, respectively, the *source* and *target* context for the tranformation at hand. The objects returned by modifiers (in this case, scaleFactor(c1) and scaleFactor(c)) are semantic objects and need to be dereferenced to the current context before they can be operated upon: this is achieved by invoking the method value(c1) on them. Notice that the same conversion function can be introduced in context c2; the only change required is the systematic replacement of all references to c1 by c2.

The conversion functions defined for semantic objects are invoked when the semantic objects are exchanged between different contexts. For example, the value of the semantic object f_r1_revenue("NTT") in context c2 is given by

```
f_r1_revenue("NTT")[value(c2) → v] ←
    f_r1_revenue("NTT")[cvt(c2) → v].
```

The method cvt(c2) can in turn be rewritten as a series of invocations on the conversion function defined on each modifier pertaining to the semantic type. Thus, in the case of moneyAmt, we would have

```
f_r1_revenue("NTT")[cvt(c2) → w] ←
    f_r1_revenue("NTT")[value(c1) → u],
    f_r1_revenue("NTT")[cvt(currency,c2)@c1,u → v],
    f_r1_revenue("NTT")[cvt(scaleFactor,c2)@c1,v → w].
```

Hence, if the conversion function for currency returns the value 9600, this will be rewritten to 9600000 by the scale-factor conversion function and returned as the value of the semantic object f_r1_revenue("NTT") in context c2.

In the same way whereby r4 is used in the assignment of values to modifiers, ancillary data sources may be used for defining appropriate conversion functions. For instance, currency conversion in context c2 is supported by the relation r3, which provides the exchange rate between two different currencies. In general, the use of ancillary data sources in context axioms will lead to the introduction of additional table lookups in the mediated query, as we have shown earlier in Section 2.

3.4 Query Mediation as Abductive Inferences

The goal of the Context Interchange framework is to provide a formal, logical basis that allows for the automatic mediation of queries such as those described in Section 2. The logical inferences which we have adopted for this purpose can be characterized as *abduction* [Kakas et al. 1993]: in the simplest case, this takes the form

> From observing A and the axiom $B \rightarrow A$
> Infer B as a possible "explanation" of A.

Abductive logic programming (ALP) [Kakas et al. 1993] is an extension of *logic programming* [Lloyd 1987] to support abductive reasoning. Specifically, an *abductive framework* [Eshghi and Kowalski 1989] is a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{I} \rangle$ where $\mathcal{T}$ is a theory, $\mathcal{I}$ is a set of integrity constraints, and $\mathcal{A}$ is a set of predicate symbols, called *abducible* predicates. Given an abductive framework $\langle \mathcal{T}, \mathcal{A}, \mathcal{I} \rangle$ and a sentence $\exists \vec{X} q(\vec{X})$ (the *observation*), the *abductive task* can be characterized as the problem of finding a *substitution* θ and a set of abducibles Δ, called the *abductive explanation* for the given observation, such that

(1) $\mathcal{T} \cup \Delta \models \forall(q(\vec{X})\theta)$,

(2) $\mathcal{T} \cup \Delta$ satisfies $\mathcal{I}$, and

(3) Δ has some properties that make it "interesting."

Requirement (1) states that Δ, together with $\mathcal{T}$, must be capable of providing an explanation for the observation $\forall(q(\vec{X})\theta)$. The prefix "$\forall$" suggests that all free variables after the substitution are assumed to be universally quantified. The consistency requirement in (2) distinguishes abductive explanations from inductive generalizations. Finally, in the characterization of Δ in (3), "interesting" means primarily that literals in Δ are atoms formed from abducible predicates: where there is no ambiguity, we refer to these atoms also as *abducibles*. In most instances, we would like Δ to also be minimal or nonredundant.

The Context Interchange framework is mapped to an abductive framework $\langle \mathcal{T}, \mathcal{A}, \mathcal{I} \rangle$ in a straightforward manner. Specifically, the domain model axioms, the elevation axioms, and the context axioms are rewritten to normal Horn clauses where nonmonotonic inheritance is simulated through the use of negation. The procedure and semantics for this transformation have been described in Abiteboul et al. [1993]. The resulting set of clauses, together with a handful of generic axioms, defines the theory $\mathcal{T}$ for the abductive framework. The integrity constraints in $\mathcal{I}$ consist of all the integrity constraints defined on the sources complemented with Clark's

Free Equality Axioms[1] [Clark 1978]. Finally, the set of abducibles $\mathcal{A}$ consists of all extensional predicates (relation names exported by sources) and references to externally stored procedures (referenced by some conversion functions).

As we have noted in Section 2, queries in a Context Interchange system are formulated under the assumption that there are no conflicts between sources and/or the receiver. Given an SQL query, context mediation is bootstrapped by tranforming this user query into an equivalent query in the internal COINL representation. For example, the query Q1 (in Section 2) will be rewritten to the following form:

$$\begin{aligned}\text{Q1}^{*}\text{:}\ & \leftarrow ans(x, y).\\ & ans(x, y) \leftarrow \texttt{r1}(x, y, \_),\ \texttt{r2}(x, z),\ y > z.\end{aligned}$$

The predicate *ans* is introduced so that only those attributes which are needed are projected as part of the answer. This translation is obviously a trivial exercise, since both COINL and relational query languages are variants of predicate calculus.

The preceding query however continues to make reference to primitive objects and (extensional) relations defined on them. To allow us to reason with the different representations built into semantic objects, we introduce two further artifacts which facilitates the systematic rewriting of a query to a form which the context mediator can work with.

—For every extensional relation $\texttt{r}$, we introduce a corresponding *semantic relation* $\bar{\texttt{r}}$ which is isomorphic to the original relation, with each primitive object in the extensional relation being replaced by its semantic object counterpart. For example, the semantic relation for $\bar{\texttt{r}}\texttt{1}$ is defined via the axiom

$$\bar{\texttt{r}}\texttt{1}(\texttt{f\_r1\_cname}(x), \texttt{f\_r1\_revenue}(x), \texttt{f\_r1\_country}(x)) \leftarrow \texttt{r1}(x, \_, \ \_).$$

A sample tuple of this semantic relation can be seen in Figure 3.

—To take into account the fact that the same semantic object may have different representations in different contexts, we enlarge the notion of classical "relational" comparison operators and insist that such comparisons are only meaningful when they are performed with respect to a given context. Formally, if $\diamond$ is some element of the set $\{=,\neq,\leq,\geq,<,>,\ldots\}$ and x, y are primitive objects or semantic objects (not necessarily of the same semantic type), then we say that

$$x \overset{c}{\diamond} y \text{ iff } (x\,[\texttt{value}(c) \rightarrow u] \text{ and } y\,[\texttt{value}(c) \rightarrow v] \text{ and } u \diamond v)$$

(In the case where both x and y are primitive objects, semantic comparison degenerates to normal relational operations, since the value of a

[1] These consist of the axioms $X = X$ (reflexivity), $X = Z \leftarrow X = Y \wedge Y = Z$ (transitivity), and inequality axioms of the type $a \neq b$, $b \neq c$ for any two non-Skolem terms which do not unify.

primitive object is given by its OID.) The intuition underlying this fabrication is best grasped through an example: in the case of f_r1_revenue(“NTT”), we know that

$$\texttt{f\_r1\_revenue(“NTT”)}\ [\texttt{value(c1)} \rightarrow \texttt{1000000}].$$

Thus, the statement f_r1_revenue(“NTT”) $\overset{c}{<}$ 5000000 is true if c = c1 but not if c = c2 (since f_r1_revenue(“NTT”)[value(c2) → 9600000]).

Using the above definitions, the context mediator can rewrite the query Q1* shown earlier to the following:

$$ans(u, v) \leftarrow \bar{\texttt{r}}\texttt{1}(x, y, \_), \bar{\texttt{r}}\texttt{2}(w, z), x \overset{c2}{=} w, y \overset{c2}{>} z, x[\texttt{value(c2)} \rightarrow u],$$
$$y[\texttt{value(c2)} \rightarrow v].$$

This is obtained by systematic renaming of each extensional predicate (r) to its semantic counterpart ($\bar{\texttt{r}}$), by replacing all comparisons (including implicit “joins”) with semantic comparisons, and making sure that attributes which are to be projected in a query correspond to the values of semantic objects in the context associated with the query.

The abductive answer corresponding to the above query can be obtained via backward chaining, using a procedure not unlike the standard *SLD-resolution* procedure [Eshghi and Kowalski 1989]. We present the intuition of this procedure below by visiting briefly the sequence of reasoning in the example query. A formal description of this procedure can be found in Bressan et al. [1997b].

Starting from the query above and resolving each literal with the theory $\mathcal{T}$ in a depth-first manner, we would have obtained the following:

$$\leftarrow \texttt{r1}(u_0, v_0, \_), \bar{\texttt{r}}\texttt{2}(w, z), \texttt{f\_r1\_cname}(u_0) \overset{c2}{=} w, \texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} z,$$
$$\texttt{f\_r1\_cname}(u_0)\ [\texttt{value(c2)} \rightarrow u], \texttt{f\_r1\_revenue}(u_0)\ [\texttt{value(c2)} \rightarrow v].$$

The subgoal $\texttt{r1}(u_0, v_0, \_)$ cannot be further evaluated and will be abducted at this point, yielding the following sequence:

$$\leftarrow \bar{\texttt{r}}\texttt{2}(w, z), \texttt{f\_r1\_cname}(u_0) \overset{c2}{=} w, \texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} z,$$
$$\texttt{f\_r1\_cname}(u_0)\ [\texttt{value(c2)} \rightarrow u], \texttt{f\_r1\_revenue}(u_0)\ [\texttt{value(c2)} \rightarrow v].$$
$$\leftarrow \texttt{r2}(u', v'), \texttt{f\_r1\_cname}(u_0) \overset{c2}{=} \texttt{f\_r2\_cname}(u'),$$
$$\texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} \texttt{f\_r2\_expenses}(u'),$$
$$\texttt{f\_r1\_cname}(u_0)\ [\texttt{value(c2)} \rightarrow u], \texttt{f\_r1\_revenue}(u_0)\ [\texttt{value(c2)} \rightarrow v].$$

Again, $\texttt{r2}(u', v')$ is abducted to yield

$$\leftarrow \texttt{f\_r1\_cname}(u_0) \overset{c2}{=} \texttt{f\_r2\_cname}(u'),$$
$$\texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} \texttt{f\_r2\_expenses}(u'),$$
$$\texttt{f\_r1\_cname}(u_0)\ [\texttt{value(c2)} \rightarrow u], \texttt{f\_r1\_revenue}(u_0)\ [\texttt{value(c2)} \rightarrow v].$$

Since companyName has no modifiers, there is no conversion function defined on instances of companyName, so the value of $\texttt{f\_r1\_cname}(u_0)$ does not vary across any context. Hence, the subgoal $\texttt{f\_r1\_cname}(u_0) \overset{c2}{=} \texttt{f\_r2\_cname}(u')$ can be reduced to just $u_0 = u'$ which unifies the variables u and u', reducing the goal further to

$$\leftarrow \texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} \texttt{f\_r2\_expenses}(u_0),$$
$$\texttt{f\_r1\_cname}(u_0)\ [\texttt{value(c2)} \rightarrow u],\ \texttt{f\_r1\_revenue}(u_0)\ [\texttt{value(c2)} \rightarrow v].$$

This process goes on until this goal list has been reduced to the empty clause. Upon backtracking, alternative abductive answers can be obtained. In this example, we obtain the following abductive answers in direct correspondance to the mediated query MQ1 shown earlier:

$$\Delta_1 = \{\ \texttt{r1}(u, v, \_),\ \texttt{r2}(u, v'),\ \texttt{r4}(u, \text{“USD”}),\ v > v'\}$$
$$\Delta_2 = \{\ \texttt{r1}(u, v_0, \_),\ \texttt{r2}(u, v'),\ \texttt{r4}(u, \text{“JPY”}),\ \texttt{r3}(\text{“JPY”}, \text{“USD”}, r),\ v = v_0 * r * 1000,\ v > v'\}$$
$$\Delta_3 = \{\ \texttt{r1}(u, v_0, \_),\ \texttt{r2}(u, v'),\ \texttt{r4}(u, y),\ y \neq \text{“USD”},\ y \neq \text{“JPY”},\ \texttt{r3}(y, \text{“USD”}, r),\ v = v_0 * r,\ v > v'\}$$

The query-rewriting technique described above may also be understood as a form of *partial evaluation*, in which a high-level specification is transformed into a lower-level program which can be executed more efficiently. In this context, the context mediator plays the role of a meta-interpreter that evaluates part of the query (identifying potential conflicts and methods for their resolution in consultation with the logic theory $\mathcal{T}$), while delaying other parts of the query that involve access to extensional databases and evaluable predicates. This compilation can be performed online or offline, i.e., at the time a query is being submitted, or in the form of precompiled view definitions that are regularly queried by users and other client applications.

4. COMPARISON WITH EXISTING APPROACHES

In an earlier report [Goh et al. 1994], we have made detailed comments on the many features that the Context Interchange approach has over traditional *loose-* and *tight-coupling* approaches. In summary, although tightly coupled systems provide better support for data access to heterogeneous systems (compared to loosely coupled systems), they do not scale-up effectively given the complexity involved in constructing a shared schema for a large number of systems and are generally unresponsive to changes for the same reason. Loosely coupled systems, on the other hand, require little central administration but are equally nonviable, since they require users to have intimate knowledge of the data sources being accessed; this assumption is generally nontenable when the number of systems involved is large and when changes are frequent.[2] The Context Interchange approach provides a novel middle ground between the two: it allows knowledge of data semantics to be independently captured in sources and receivers (in the form of context theories), while allowing a specialized

[2] We have drawn a sharp distinction between the two here to provide a contrast of their relative features. In practice, one is most likely to encounter a hybrid of the two strategies. It should however be noted that the two strategies are incongruent in their outlook and are *not* able to easily take advantage of each other's resources. For instance, data semantics encapsulated in a shared schema cannot be easily extracted by a user to assist in formulating a query which seeks to reference the source schemas directly.

mediator (the Context Mediator) to undertake the role of detecting and reconciling potential conflicts at the time a query is submitted.

At a cursory level, the Context Interchange approach may appear similar to many contemporary integration approaches. However, we posit that the similarities are superficial, and that our approach represents a significant departure from these strategies. Given the proliferation of system prototypes, it is not practical to compare our approach with each of these. The following is a sampling of contemporary systems which are representative of various alternative integration approaches.

A number of contemporary systems (e.g., Pegasus [Ahmed et al. 1991], the ECRC Multidatabase Project [Jonker and Schütz 1995], SIMS [Arens and Knoblock 1992], and DISCO [Tomasic et al. 1995]) have attempted to rejuvenate the loose- or tight-coupling approach through the adoption of an object-oriented formalism. For loosely coupled systems, this has led to more expressive data transformation (e.g., O*SQL [Litwin 1992]); in the case of tightly coupled systems, this helps to mitigate the effects of complexity in schema creation and change management through the use of abstraction and encapsulation mechanisms. Although the Context Interchange strategy embraces "object orientation" for the same reasons, it differs by not requiring pairwise reconciliation of semantic conflicts to be incorporated as part of the shared schema. For instance, our approach does not require the domain model to be updated each time a new source is added; this is unlike tightly coupled systems where the shared schema needs to be updated by-hand each time such an event occurs, even when conflicts introduced by the new source are identical to those which are already present in existing sources. Yet another difference is that although a deductive object-oriented formalism is also used in the Context Interchange approach, "semantic objects" in our case exist only conceptually and are never actually materialized during query evaluation. Thus, unlike some other systems (e.g., the ECRC prototype), we do not require an intermediary "object store" where objects are instantiated before they can be processed. In our implementation, both user queries and their mediated counterpart are relational. The mediated query can therefore be executed by a classical relational DBMS without the need to reinvent a query-processing subsystem.

In the Carnot system [Collet et al. 1991], semantic interoperability is accomplished by writing *articulation axioms* which translate "statements" which are true in individual sources to statements which are meaningful in the Cyc knowledge base [Lenat and Guha 1989]. A similar approach is adopted in Faquhar et al. [1995], where it is suggested that domain-specific *ontologies* [Gruber 1991], which may provide additional leverage by allowing the ontologies to be shared and reused, can be used in place of Cyc. While we like the explicit treatment of contexts in these efforts and share their concern for sustaining an infrastructure for data integration, our realization of these differs in several important ways. First, our domain model is a much more impoverished collection of rich types compared to the richness of the Cyc knowledge base. Simplicity is a feature here because the construction of a rich and complex shared model is laborious and error prone, not to mention that it is almost impossible to maintain. Second, the translation of sentences from one context to another is embedded in axioms present in individual context theories, and are not part of the domain model. This means that there is greater scope for different users to introduce conversion functions which are most appropriate for their purposes without requiring these differences to be accounted for globally. Finally, semantics of data is represented in an "object-centric" manner as opposed to a "sentential" representation. For example, to relate two statements (σ and σ') in different distinct contexts c and c', a lifting axiom of the form

$$ist(c, \sigma) \Leftrightarrow ist(c', \sigma')$$

will have to be introduced in Cyc. In the Context Interchange approach, we have opted for a "type-based" representation where conversion functions are attached to types in different contexts. This mechanism allows for greater sharing and reuse of semantic encoding. For example, the same type may appear many times in different predicates (e.g., consider the type `moneyAmt` in a financial application). Rather than writing a lifting axiom for each predicate that redundantly describes how different reporting currencies are resolved, we can simply associate the conversion function with the type `moneyAmt`.

Finally, we remark that the TSIMMIS [Papakonstantinou et al. 1995; Quass et al. 1995] approach stems from the premise that information integration could not, and should not, be fully automated. With this in mind, TSIMMIS opted in favor of providing both a framework and a collection of tools to assist humans in their information processing and integration activities. This motivated the invention of a "lightweight" object model which is intended to be *self-describing*. For practical purposes, this translates to the strategy of making sure that attribute labels are as descriptive as possible and opting for free-text descriptions ("man-pages") which provide elaborations on the semantics of information encapsulated in each object. We concur that this approach may be effective when the data sources are ill structured and when consensus on a shared vocabulary cannot be achieved. However, there are also many other situations (e.g., where data sources are relatively well structured and where *some* consensus can be reached) where human intervention is not appropriate or necessary: this distinction is primarily responsible for the different approaches taken in TSIMMIS and our strategy.

5. CONCLUSION

Although there had been previous attempts at formalizing the Context Interchange strategy (see, for instance, Sciore et al. [1994]), a tight integration of the representational and reasoning formalisms has been consistently lacking. This article has filled this gap by introducing a well-founded logical framework for capturing context knowledge and in demonstrating that query mediation can be formally understood with reference to current work in abductive logic programming. The advancements made in this theoretical frontier have been instrumental in the development of a prototype which provides for the integration of data from disparate sources accessible on the Internet. The architecture and features of this prototype have been reported in Bressan et al. [1997a] and will not be repeated here due to space constraints.

The adoption of a declarative encoding of data semantics brings about other side benefits, chief among which is the ability to query directly the semantics of data which are implicit in different systems. Consider, for instance, the query formulated on the motivational example introduced earlier in the article, that is based on a superset of SQL:[3]

```
Q2: SELECT r1.cname, r1.revenue.scaleFactor IN c1,
           r1.revenue.scaleFactor IN c2 FROM r1
    WHERE  r1.revenue.scaleFactor IN c1
           <> r1.revenue.scaleFactor IN c2;
```

Intuitively, this query asks for companies for which scale-factors for reporting "revenue" in `r1` (in context c1) differ from that which the user assumes (in context c2). We refer to queries such as Q2 as *knowledge-level queries*, as opposed to *data-level queries* which are enquires on factual data present in data sources. Knowledge-level queries have received little attention in the database literature and to our knowledge have not been addressed by the data integration community. This is a significant gap in the literature given that heterogeneity in disparate data sources arises primarily from incompatible assumptions about how data are interpreted. Our ability to integrate access to both data and semantics can be exploited by users to gain insights into differences among particular systems; for example, we may want to know "Do sources A and B report a piece of data differently? If so, how?" Alternatively, this facility may be exploited by a query optimizer which may want to identify sites with minimal conflicting interpretations in identifying a query plan which requires less costly data transformations.

Interestingly, knowledge-level queries can be answered using the exact same inference mechanism for mediating data-level queries. Hence, submitting query Q2 to the Context Mediator will yield the result

```
MQ2: SELECT r1.cname, 1000, 1 FROM r1, r4
     WHERE r1.country = r4.country AND r4.currency = 'JPY';
```

which indicates that the answer consists of companies for which the reporting currency attribute is `'JPY'`, in which case the scale-factors in context c1 and c2 are `1000` and `1` respectively. If desired, the mediated query MQ2 can be evaluated on the extensional data set to return an answer grounded in the extensional data set. Hence, if MQ2 is evaluated on

[3] Sciore et al. [1992] have described a similar (but not identical) extension of SQL in which context is treated as a "first-class object." We are not concerned with the exact syntax of such a language here; the issue at hand is how we might support the underlying inferences needed to answer such queries.

the data set shown in Figure 1, we would obtain the singleton answer ⟨'NTT', 1000, 1⟩.

Yet another feature of Context Interchange is that *answers* to queries can be both intensional and extensional. Extensional answers correspond to fact sets which one normally expects of a database retrieval. Intensional answers, on the other hand, provide only a characterization of the extensional answers *without* actually retrieving data from the data sources. In the preceding example, MQ2 can in fact be understood as an intensional answer for Q2, while the tuple obtained by the evaluation of MQ2 constitutes the extensional answer for Q2.

As seen from the above example, intensional answers are grounded in extensional predicates (i.e., names of relations), evaluable predicates (e.g., arithmetic operators or "relational" operators), and external functions which can be directly evaluated through system calls. The intensional answer is thus no different from a query which can normally be evaluated on a conventional query subsystem of a DBMS. Query answering in a Context Interchange system is thus a two-step process: an intensional answer is first returned in response to a user query; this can then be executed on a conventional query subsystem to obtain the extensional answer.

The intermediary intensional answer serves a number of purposes [Imielinski 1987]. Conceptually, it constitutes the mediated query corresponding to a user query and can be used to confirm the user's understanding of what the query actually entails. More often than not, the intensional answer can be more informative and easier to comprehend compared to the extensional answer it derives. (For example, the intensional answer MQ2 actually conveys more information than merely the extensional answer comprising a single tuple.) From an operational standpoint, the computation of extensional answers is likely to be many orders of magnitude more expensive compared to the evaluation of the corresponding intensional answer. It therefore makes good sense not to continue with query evaluation if the intensional answer satisfies the user. From a practical standpoint, this two-stage process allows us to separate query mediation from query optimization and execution. As we have illustrated in this article, query mediation is driven by logical inferences which do not bond well with (predominantly cost-based) optimization techniques that have been developed [Mumick and Pirahesh 1994; Seshadri et al. 1996]. The advantage of keeping the two tasks apart is thus not merely a conceptual convenience, but allows us to take advantage of mature techniques for query optimization in determining how best a query can be evaluated.

To the best of our knowledge, the application of abductive reasoning to "database problems" has been confined to the view-update problem [Kakas and Mancarella 1990]. Our use of abduction for query rewriting represents a potentially interesting avenue which warrants further investigation. For example, consistency checking performed in the abduction procedure allows a mediated query to be pruned to arrive at intensional answers which are more comprehensible as well as queries which are more efficient. This bears some similarity to techniques developed for *semantic query optimization* [Chakravarthy et al. 1990] and appears to be useful for certain types of optimization problems.

ACKNOWLEDGMENTS

We like to thank Umesh Dayal, Michael Kifer, Arnie Rosenthal, Edward Sciore, and the anonymous referees for their feedback on earlier drafts of this article.

In Memory of Cheng Hian Goh (1965–1999)

Cheng Hian Goh passed away on April 1, 1999, at the age of 33. He is survived by his wife Soh Mui Lee and two sons, Emmanuel and Gabriel, to whom we all send our deepest condolences.

Cheng Hian received his PhD in Information Technologies from the Massachusetts Institute of Technology in February, 1997. He joined the Department of Computer Science, National University of Singapore as an Assistant Professor in November, 1996.

He loved and was dedicated to his works—teaching as well as research. He believed in giving his students the best and what they deserved. As a young database researcher, he had made major contributions to the field as testified by his publications (in ICDE'97, VLDB'98, and ICDE'99).

Cheng Hian was a very sociable person, and often sought the company of friends. As such, he was loved by those who come in contact with him. He would often go the extra mile to help his friends and colleagues. He was a great person, and had touched the lives of many. We suddenly realized that there are many things that we will never do together again. We will miss him sorely, and his laughter, and his smile...

REFERENCES

Abiteboul, S., Lausen, G., Uphoff, H., and Waller, E. 1993. Methods and rules. *SIGMOD Rec. 22*, 2 (June 1, 1993), 32–41.

Ahmed, R., De Smedt, P., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Rafii, A., and Shan, M.-C. 1991. The Pegasus heterogeneous multidatabase system. *IEEE Comput. 24*, 12 (Dec. 1991), 19–27.

Arens, Y. and Knoblock, C. A. 1992. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (CIKM-92, Baltimore, MD, Nov.), Y. Yesha, Ed. 92–101.

Bressan, S., Goh, C. H., Fynn, K., Jakobisiak, M., Hussein, K., Kon, H., Lee, T., Madnick, S., Pena, T., Qu, J., Shum, A., and Siegel, M. 1997a. The Context Interchange mediator prototype. *SIGMOD Rec. 26*, 2, 525–527.

Bressan, S., Goh, C. H., Lee, T., Madnick, S., and Siegel, M. 1997b. A procedure for mediation of queries to sources in disparate contexts. In *Proceedings of the 1997 international symposium on Logic programming* (ILPS '97, Port Washington, NY, Oct. 12–16, 1997), J. Małuszyński, I. V. Ramakrishnan, and T. Swift, Eds. MIT Press, Cambridge, MA, 213–227.

Catarci, T. and Lenzerini, M. 1993. Representing and using interschema knowledge in cooperative information systems. *Int. J. Intell. Coop. Inf. Syst. 2*, 4 (Dec.), 375–399.

Chakravarthy, U. S., Grant, J., and Minker, J. 1990. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst. 15*, 2 (June 1990), 162–207.

CLARK, K. 1978. Negation as failure. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, NY, 292–322.

COLLET, C., HUHNS, M. N., AND SHEN, W.-M. 1991. Resource integration using a large knowledge base in Carnot. *IEEE Comput. 24*, 12 (Dec. 1991), 55–62.

DOBBIE, G. AND TOPOR, R. 1995. On the declarative and procedural semantics of deductive object-oriented systems. *J. Intell. Inf. Syst. 4*, 2 (Mar. 1995), 193–219.

ESHGHI, K. AND KOWLASKI, R. A. 1989. Abduction compared with negation by failure. In *Proceedings of the 6th International Conference on Logic Programming* (Lisbon, Spain). 234–255.

FAQUHAR, A., DAPPERT, A., FILKES, R., AND PRATT, W. 1995. Integrating information sources using context logic. In *Proceedings of the AAAI-95 Spring Symposium on Information Gathering from Distributed Heterogeneous Environments*. AAAI Press, Menlo Park, CA.

GOH, C. H., BRESSAN, S., MADNICK, S. E., AND SIEGAL, M. D. 1996. Context interchange: Representing and reasoning about data semantics in heterogeneous systems. Sloan School Working Paper No. 3928. MIT-Alfred P. Sloan School of Management, Cambridge, MA.

GOH, C. H., MADNICK, S. E., AND SIEGEL, M. D. 1994. Context interchange: Overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *Proceedings of the 3rd International Conference on Information and Knowledge Management* (CIKM '94, Gaithersburg, Maryland, Nov. 29–Dec. 2, 1994), N. R. Adam, B. K. Bhargava, and Y. Yesha, Eds. ACM Press, New York, NY, 337–346.

GRUBER, T. R. 1991. The role of common ontology in achieving sharable, reusable knowledge bases. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference* (Cambridge, MA.), J. A. Allen, R. Files, and E. Sandewall, Eds. Morgan Kaufmann, San Mateo, California, 601–602.

HULL, R. 1997. Managing semantic heterogeneity in databases: a theoretical prospective. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (PODS '97, Tucson, Arizona, May 12–14, 1997), A. Mendelzon and Z. M. Özsoyoglu, Eds. ACM Press, New York, NY, 51–61.

IMEILINSKI, T. 1987. Intelligent query answering in rule based systems. *J. Logic Program. 4*, 3 (Sept. 1987), 229–257.

JONKER, W. AND SCHÜTZ, H. 1995. The ECRC multi database system. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (SIGMOD '95, San Jose, CA, May 23–25, 1995), M. Carey and D. Schneider, Eds. ACM Press, New York, NY, 490.

KAKAS, A. C. AND MANCARELLA, P. 1990. Database updates through abduction. In *Proceedings of the 16th International Conference on Very Large Databases* (Brisbane, Australia, Aug. 13–16, 1990), D. McLeod, R. Sacks-Davis, and H. Schek, Eds. Morgan Kaufmann Publishers Inc., San Francisco, CA, 650–661.

KAKAS, A. C., KOWALSKI, R. A., AND TONI, F. 1993. Abductive logic programming. *J. Logic Program. 2*, 6, 719–770.

KIFER, M., LAUSEN, G., AND WU, J. 1995. Logical foundations of object-oriented and frame-based languages. *J. ACM 42*, 4 (July 1995), 741–843.

KUHN, E. AND LUDWIG, T. 1988. VIP-MDBS: A logic multidatabase system. In *International Symposium on Databases in Parallel and Distributed Systems* (Austin, Texas, Dec. 5-7, 1988), J. E. Urban, Ed. IEEE Computer Society Press, Los Alamitos, CA, 190–201.

LANDERS, T. AND ROSENBERG, R. 1982. An overview of Multibase. In *Proceedings of the 2nd International Symposium for Distributed Databases*. 153–183.

LENAT, D. B. AND GUHA, R. V. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley Publishing Co., Inc., Redwood City, CA.

LEVY, A. Y., MENDELZON, A. O., AND SAGIV, Y. 1995a. Answering queries using views (extended abstract). In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (PODS '95, San Jose, California, May 22–25, 1995), M. Yannakakis, Ed. ACM Press, New York, NY, 95–104.

LEVY, A. Y., SRIVASTAVA, D., AND KIRK, T. 1995b. Data model and query evaluation in global information systems. *J. Intell. Inf. Syst. 5*, 2 (Sept. 1995), 121–143.

LITWIN, W. 1992. O*SQL: A language for object oriented multidatabase interoperability. In *Proceedings of the Conference on IFIP WG2.6 Database Semantics and Interoperable Database Systems (DS-5)* (Lorne, Victoria, Australia), D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, Eds. North-Holland Publishing Co., Amsterdam, The Netherlands, 119–138.

LITWIN, W. AND ABDELLATIF, A. 1987. An overview of the multi-database manipulation language MDSL. *Proc. IEEE 75*, 5, 621–632.

LLOYD, J. W. 1987. *Foundations of Logic Programming*. 2nd ed. Springer-Verlag Symbolic Computation and Artificial Intelligence Series. Springer-Verlag, Vienna, Austria.

MCCARTHY, J. 1987. Generality in artificial intelligence. *Commun. ACM 30*, 12 (Dec. 1987), 1030–1035.

MUMICK, I. S. AND PIRAHESH, H. 1994. Implementation of magic-sets in a relational database system. *SIGMOD Rec. 23*, 2 (June 1994), 103–114.

PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., AND WIDOM, J. 1995. Object exchange across heterogeneous information sources. In *Proceedings of the IEEE International Conference on Data Engineering* (Mar.). IEEE Press, Piscataway, NJ.

QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., AND WIDON, J. 1995. Querying semistructured heterogeneous information. In *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases* (Singapore, Dec.). Springer-Verlag, Berlin, Germany.

SCIORE, E., SIEGAL, M., AND ROSENTHAL, A. 1992. Context interchange using meta-attributes. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (CIKM-92, Baltimore, MD, Nov.), Y. Yesha, Ed. 377–386.

SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. 1994. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Trans. Database Syst. 19*, 2 (June 1994), 254–290.

SESHADRI, P., HELLERSTEIN, J. M., PIRAHESH, H., LEUNG, T. C., RAMAKRISHNAN, R., SRIVASTAVA, D., STUCKEY, P. J., AND SUDARSHAN, S. 1996. Cost-based optimization for magic: Algebra and implementation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (SIGMOD '96, Montreal, Canada). ACM, New York, NY, 435–446.

SHETH, A. P. AND LARSON, J. A. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv. 22*, 3 (Sept. 1990), 183–236.

SIEGEL, M. AND MADNICK, S. E. 1991. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th Conference on Very Large Data Bases* (Barcelona, Spain, Sept.). VLDB Endowment, Berkeley, CA, 133–145.

TEMPLETON, M., BRILL, D., DAO, S. K., LUND, E., WARD, P., CHEN, A. L. P., AND MACGREGOR, R. 1987. Mermaid—a front end to distributed heterogeneous databases. *Proc. IEEE 75*, 5, 695–708.

TOMASIC, A., RASCHID, L., AND VALDURIEZ, P. 1996. Scaling heterogeneous databases and the design of DISCO. In *Proceedings of the 16th IEEE International Conference on Distributed Computing Systems* (Hong Kong, May). IEEE Computer Society Press, Los Alamitos, CA.

ULLMAN, J. D. 1997. Information integration using logical views. In *Proceedings of the 6th International Conference on Database Theory* (ICDT '97, Delphi, Greece, Jan.). Springer-Verlag, Berlin, Germany, 19–40.

WIEDERHOLD, G. 1992. Mediators in the architecture of future information systems. *IEEE Comput. 25*, 3 (Mar. 1992), 38–49.

Received: June 1997; revised: April 1998 and June 1998; accepted: August 1998