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

Cheng Hian Goh, Stephane Bressan,
Stuart Madnick, Michael Siegel

Sloan WP# 3941   CISL WP# 97-03
February 1997

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142

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

Cheng Hian Goh
Dept of Info Sys & Comp Sc
National University of S'pore
gohch@iscs.nus.sg

Stéphane Bressan Stuart Madnick Michael Siegel
Sloan Sch of Mgt
Massachusetts Institute of Technology
{sbressan,smadnick,msiegel}@mit.edu

Abstract

The COntext INterchange (COIN) 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 paper, we illustrate the new features which are made possible by this integration strategy, such as allowing both queries on shared views and export schemas to be mediated using the same framework, supporting both knowledge-level as well as data-level queries, and furnishing both intensional and extensional answers to user queries. We then proceed to describe the COIN framework, which provides a formal characterization of the representation and reasoning underlying the Context Interchange strategy. This formalization constitutes a well-founded basis for the design and implementation of a prototype embodying the features which we have described.

Keywords: Context, heterogeneous databases, logic and databases, mediators, semantic interoperability.

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

*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), and the PROductivity From Information Technology (PROFIT) project at MIT.

for VLDB '97

ability to exchange information meaningfully). This problem is sometimes referred to as the need for *semantic interoperability* [33] among *autonomous* and *heterogeneous* systems.

The *Context Interchange* (COIN) strategy [34; 31] is a mediator-based approach [37] 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.

These insights are novel because they depart from classical 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 [24], VIP-MDBMS [20]), 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 [21], Mermaid [35]).

Our goal in this paper is (1) to illustrate various novel features of the COIN 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 paper are new (with respect to our previous works) and represent a significant departure from integration approaches which have been described in the literature. Unlike most existing research which addresses exclusively the issue of interoperable data exchanges, the proposed integration strategy supports both "knowledge-level" as well as "data-level" queries, and is capable of returning both "extensional" as well as "intensional" answers. In addition, both "multidatabase" queries as well as queries on "shared views" can be mediated while allowing semantic descriptions of disparate sources to remained loosely-coupled to one another. This strategy has also been validated in a prototype system which provides access to both traditional data sources (e.g., Oracle data systems) as well as semi-structured information sources (e.g., Web-sites).

The rest of this paper is organized as follows. Following this introduction, we present a motivational example which is used to highlight selected features of the Context Interchange strategy. Section 3 describes the COIN 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 contribution and describe 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 paper presenting more of the technical details is available as a working paper [12]. A report on the Prototype can also be found in [4]. An in-depth discussion of the context mediation procedure and its implementation can be found in a companion paper [5].

2 Context Interchange by Example

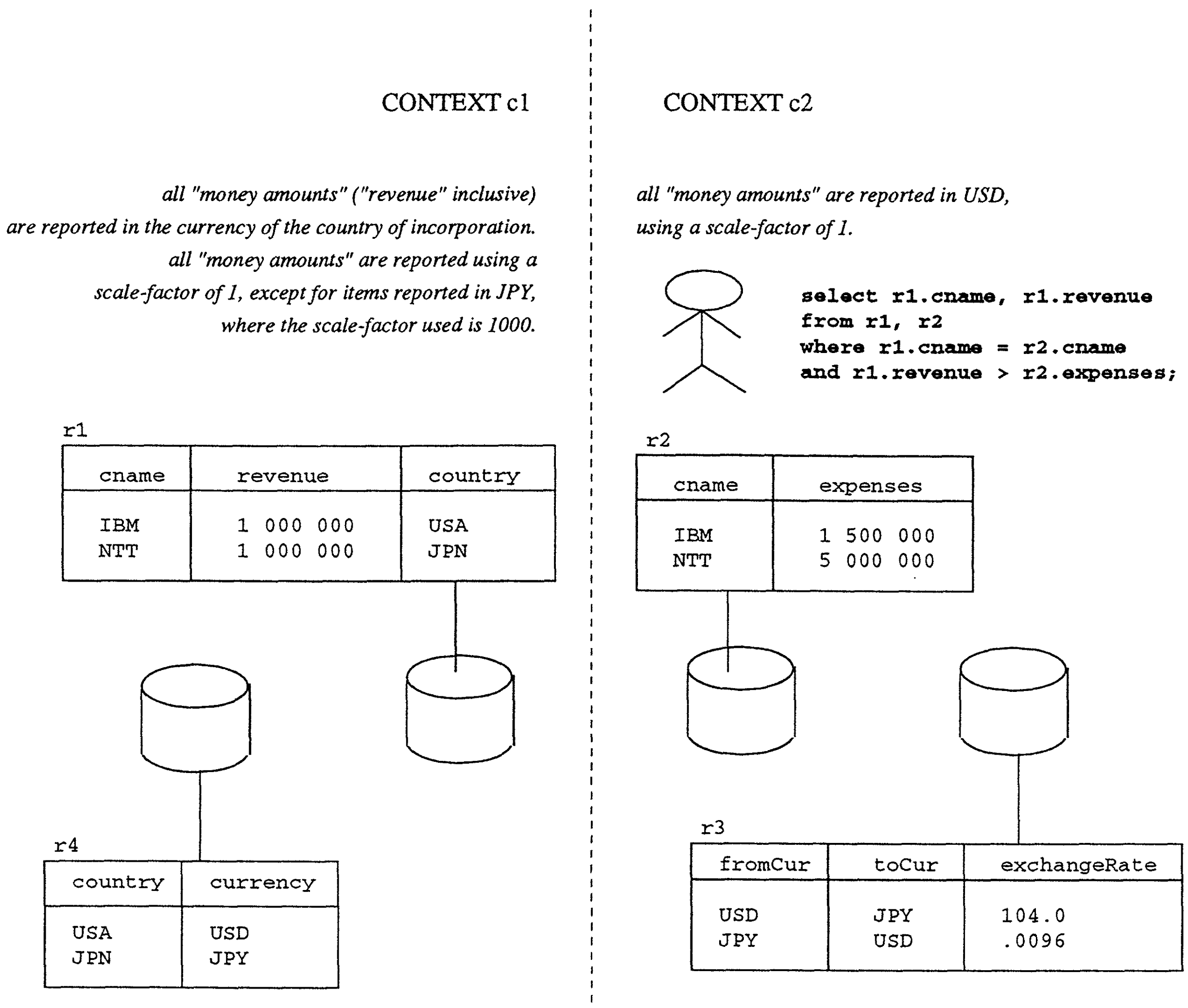
Consider the scenario shown in Figure 1, deliberately kept simple for didactical reasons. Data on "revenue" and "expenses" (respectively) for some collection of companies are available in two autonomously-administered data sources, each comprised of a single relation[1], denoted by r1 and r2 respectively. 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 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 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

[1]Throughout this paper, we make the assumption that the relational data model is adopted to be the *canonical data model* [33]: i.e., we assume that the database schemas exported by the sources are relational and that queries are formulated using SQL (or some extension thereof). This simplifies the discussion by allowing us to focus on semantic conflicts in disparate systems without being detracted by conflicts over data model constructs. The choice of the relational data model is one of convenience rather than necessity, and is not to be construed as a constraint of the integration strategy being proposed.



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

Figure 1: Example scenario.

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 $\times$ 1,000 $\times$ 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

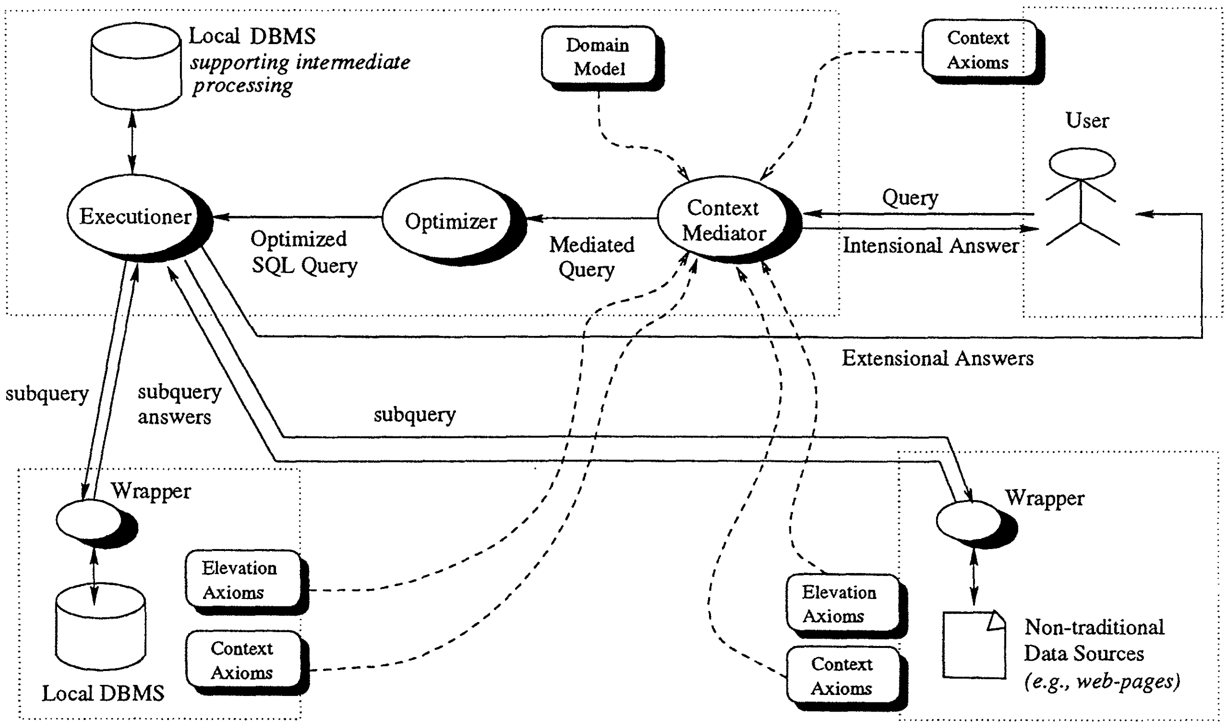


Figure 2: Architecture of a Context Interchange System

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 remainder of this section, we describe the queries and answers that can be supported by such an architecture.

2.1 Mediation of "Multidatabase" Queries

The query Q1 shown above is in fact similar to "multidatabase" MDSL queries described in [24] whereby the export schemas of individual data sources are explicitly referenced. Nonetheless, unlike the approach advocated in [24], we maintain the opinion that users should be insulated from underlying semantic heterogeneity: i.e., users should not be required to engage in detecting or reconciling potential conflicts between any two systems. In the Context Interchange system, this function is undertaken 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). This second query, 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 much flexibility in formulating a query, it also requires users to know what data are present *where* 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 achieving 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 Knowledge-Level versus Data-Level Queries

Instead of inquiring about stored data, it is sometimes useful to be able to query the semantics of data which are implicit in different systems. Consider, for instance, the query based on a superset of SQL[2]:

```
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 ("Do sources A and B report a piece of data

[2]Sciore et al. [30] have described a similar (but not identical) extension of SQL in which context is treated as a "first-class object". We are not concern 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.

differently? If so, how?"), or by a query optimizer which may want to identify sites with minimal conflicting interpretations (to minimize costs associated with 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 the data set shown in Figure 1, we would obtain the singleton answer <'NTT', 1000, 1>.

2.4 Extensional versus Intensional Answers

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.

In the COIN framework, 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 [15]. 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 of single tuple.) From an operational standpoint, the computation of extensional answers are 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 will illustrate later in this paper, query mediation is driven by logical inferences which do not bond well with (predominantly cost-based) optimization techniques that have been developed [27; 32]. 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.

3 The Context Interchange Framework

In [26], McCarthy pointed 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:

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

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", 1000000, "JPY")`).
$\bar{c}$: *ist*(`c2, r2("NTT", 5000000)`).

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 of three components:

- The *domain model* is a collection or "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 also 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 COIN 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* are 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 COIN framework is constructed on a deductive and object-oriented data model (and language) of the family of F(rame)-logic [19], 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, [1]). 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* [13]. This observation will be revisited in Section 4 where we compare our approach to the integration strategy adopted in Carnot [8].

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-*

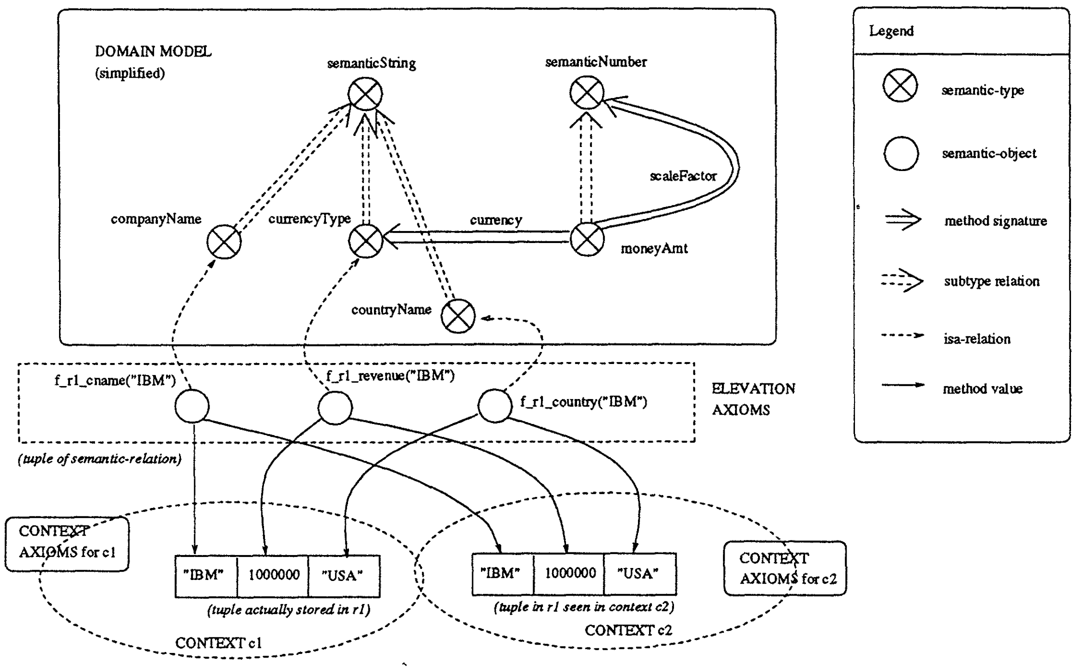


Figure 3: A graphical illustration of the different components of a Context Interchange framework.

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- 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 have 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("IBM")` 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("IBM")` is the unique oid and is invariant under all circumstances. This object however 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("IBM")[value(c1) → 1000000].
f_r1_revenue("IBM")[value(c2) → 9600000].
```

The above statements are illustrative of statements written in the coin language (COINL), which mirrors closely that of F-logic [19]. The token `value(c1)` is a *parameterized method* and is said to return the value `1000000` when invoked on the object `f_r1_revenue("IBM")`. The same statements could have been written using a predicate calculus notation:

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

The choice of an object-logic however allows certain features (e.g., inheritance and over-ridding) 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 a 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. The first introduces a semantic-object object corresponding to each attribute of a tuple in the source. For example, the statement

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

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* [25]

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

by replacing the existentially quantified variable u with the *Skolem object* [25] $\texttt{f\_r1\_revenue}(x, y, z)$. Notice that the *Skolem function* (`f_r1_revenue`) is chosen such that it uniquely identifies the relation and attribute in question. In this example, it turns out that the functional dependency `cname` $\rightarrow$ `{revenue,country}` holds on `r1`: this allows us to replace $\texttt{f\_r1\_revenue}(x, y, z)$ by $\texttt{f\_r1\_revenue}(x)$ without any loss of generality. This follows trivially from the fact that whenever we have $\texttt{f\_r1\_revenue}(x, y, z)$ and $\texttt{f\_r1\_revenue}(x, y', z')$, 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. This is easily captured in the sentence

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

Consider, for instance, the semantic-object `f_r1_revenue("IBM")` 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 define 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 complement this declarative specification by introducing the "methods" (aka *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 while using a scale-factor of 1 is made explicit in the following axioms:

$$x : \texttt{moneyAmt}, y : \texttt{semanticNumber} \vdash y\texttt{[value(c2)} \rightarrow \texttt{1]} \leftarrow x\texttt{[scaleFactor(c2)} \rightarrow y\texttt{]}.$$
$$x : \texttt{moneyAmt}, y : \texttt{currencyType} \vdash y\texttt{[value(c2)} \rightarrow \texttt{"USD"]} \leftarrow x\texttt{[currency(c2)} \rightarrow y\texttt{]}.$$

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 [9]. By making explicit the types to which axioms are attached, we are able to simulate non-monotonic inheritance through the use of negation as in [1].

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:

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

Following Prolog's convention, the token '_' is used in denoting an "anonymous" variable. In the first axiom above, r4 is assumed to be in the same context as r1 and constitute an ancillary data source for defining part of the context (in this case, the currency used in reporting moneyAmt).

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 different context with different assumptions (aka 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 the assumption 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 $\vdash$

$$x\texttt{[cvt(scaleFactor,c1)@}c, u \rightarrow v\texttt{]} \leftarrow x\texttt{[scaleFactor(c1)} \rightarrow \texttt{\_[value(c1)} \rightarrow f\texttt{]]},\ x\texttt{[scaleFactor(}c\texttt{)} \rightarrow \texttt{\_[value(c1)} \rightarrow g\texttt{]]},\ v = u * g/f.$$

In the "antecedent" of the statement above, the first literal returns the scale-factor of x in context c1. In contrasts, 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 needs to be de-referenced 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("IBM") in context c2 is given by

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

The method cvt(c2) can in turn be rewritten as a series of invocation 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("IBM")[cvt(c2) → w] ←
     f_r1_revenue("IBM")[value(c1) → u],
     f_r1_revenue("IBM")[cvt(currency,c2)@c1,u → v],
     f_r1_revenue("IBM")[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("IBM") 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 COIN framework is to provide a formal, logical basis that allows for the automatic mediation of queries embodying those features described earlier in Section 2. The logical inferences which we have adopted for this purpose is sometimes referred to as *abduction* [17]: 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) [17] is an extension of *logic programming* [25] to support abductive reasoning. Specifically, an *abductive framework* [10] is a triple $<\mathcal{T}, \mathcal{A}, \mathcal{I}>$ 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 $<\mathcal{T}, \mathcal{A}, \mathcal{I}>$ 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 makes 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 non-redundant.

The COIN framework is mapped to an abductive framework $<T, I, A>$ in a straight-forward manner. Specifically, the domain model axioms, the elevation axioms, and the context axioms are rewritten to definite Horn clauses where non-monotonic inheritance is simulated through the use of negation. The procedure and semantics for this transformation has been described in [1]. The resulting set of clauses, together with a handful of generic axioms, define the theory T for the abductive framework. The integrity constraints in I consists of all the integrity constraints defined on the sources complemented with Clark's *Free Equality Axioms* [7]. Finally, the set of abducibles 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 on page 3 will be rewritten to

$$\begin{array}{ll} \leftarrow ans(x,y). & \\ ans(x,y) \leftarrow \texttt{r1}(x,y,\_),\ \texttt{r2}(x,z),\ y > z. & \quad \ldots\ldots\ldots\ldots (\star) \end{array}$$

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 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 $\bar{\texttt{r1}}$ is defined via the axiom:

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

  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 insists 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- 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-object, semantic comparison degenerates to normal relational operations since the value of a primitive-object is given by its oid.) The intuition underlying this fabrication is best grasp through an example: in the case of f_r1_revenue("IBM"), we know that

$$\texttt{f\_r1\_revenue("IBM")[value(c1)} \rightarrow \texttt{1000000]}.$$

Thus, the statement $\texttt{f\_r1\_revenue("IBM")} \overset{c}{<} 5000000$ is true if $c =$c1 but not if $c =$c2 (since $\texttt{f\_r1\_revenue("IBM")[value(c2)} \rightarrow \texttt{9600000]}$).

Building on the above definitions, the context mediator can now rewrite the query ($\star$) shown earlier to the following:

$$ans(u,v) \leftarrow \bar{\texttt{r1}}(x,y,\_),\ \bar{\texttt{r2}}(w,z),\ x \overset{c2}{=} w,\ y \overset{c2}{>} z,\ x\texttt{[value(c2)} \rightarrow u\texttt{]},\ y\texttt{[value(c2)} \rightarrow v\texttt{]}.$$

This is obtained by systematic renaming of each extensional predicate (r) to its semantic counterpart ($\bar{\text{r}}$), 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 [10]. 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 [12].

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

$$\leftarrow \texttt{r1}(u_0,v_0,\_),\ \bar{\texttt{r2}}(w,z),\ \texttt{f\_r1\_cname}(u_0) \overset{c2}{=} w,\ \texttt{f\_r1\_revenue}(u_0) \overset{c2}{>} z,$$
$$\texttt{f\_r1\_cname}(u_0)\texttt{[value(c2)} \rightarrow u\texttt{]},\ \texttt{f\_r1\_revenue}(u_0)\texttt{[value(c2)} \rightarrow v\texttt{]}.$$

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

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

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

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

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:

$\leftarrow$ `f_r1_revenue(`u_0`)` $\overset{c2}{>}$ `f_r2_expenses(`u_0`),`
`f_r1_cname(`u_0`) [value(c2)` $\rightarrow u$`], f_r1_revenue(`u_0`)[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:

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

Notice that there is nothing inherent in the rewriting or the inferences that treats knowledge-level queries differently from extensional queries. It is possible however for modifier values to be returned as part of the substitution along with the intensional answers, as we have seen earlier on in the sample query MQ2.

4 Comparison With Existing Approaches

In an earlier report [13], 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 non-viable since they require users to have intimate knowledge of the data sources being accessed; this assumption is generally non-tenable when the number of systems involved is large and when changes are frequent[3]. 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 mediator (the Context Mediator) to undertake the role of detecting and reconciling potential conflicts at the time a query is submitted.

[3]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.

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 radical 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 [2], the ECRC Multidatabase Project [16], SIMS [3], and DISCO [36]) 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 [23]); 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 relying on human intervention in reconciling conflicts a priori in 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 re-invent a query processing subsystem.

In the Carnot system [8], 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 [22]. A similar approach is adopted in [11], where it is suggested that domain-specific *ontologies* [14], 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 differ 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 upkeep. 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 describe how different reporting currencies are resolved, we can simply associate the conversion function with the type `moneyAmt`.

Finally, we remark that the TSIMMIS [28; 29] 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 "light-weight" 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, [31]), a tight integration of the representational and reasoning formalisms has been consistently lacking. This paper 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 has 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 has been reported in [4] and will not be repeated here due to space constraints.

To the best of our knowledge, the application of abductive reasoning to "database problems" has been confined to the view-update problem [18]. Our use of abduction for query rewriting represents a potentially interesting avenue which warrants further investigation. For example, we observed that consistency checking performed in the abduction procedure may sometimes cause intensional answers to be pruned to arrive at answers which are better comprehensible and more efficient. This bears some similarity to techniques developed for *semantic query optimization* [6] and appears to be useful for certain types of optimization problems.

References

[1] Abiteboul, S., Lausen, G., Uphoff, H., and Walker, E. Methods and rules. In *Proceedings of the ACM SIGMOD Conference* (Washington, DC, May 1993), pp. 32–41.

[2] Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Raffi, A., and Shan, M.-C. The Pegasus heterogeneous multidatabase system. *IEEE Computer 24*, 12 (1991), 19–27.

[3] Arens, Y., and Knoblock, C. A. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 92–101.

[4] Bressan, S., Fynn, K., Goh, C. H., Jakobisiak, M., Hussein, K., Kon, H., Lee, T., Madnick, S., Pena, T., Qu, J., Shum, A., and Siegel, M. The COntext INterchange mediator prototype. In *Proc. ACM SIGMOD/PODS Joint Conference* (1997). To appear.

[5] Bressan, S., Goh, C. H., Madnick, S., and Siegel, M. A procedure for context mediation of queries to disparate sources. submitted for publication, Feb 1997.

[6] Chakravarthy, U. S., Grant, J., and Minker, J. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems 15*, 2 (1990), 162–207.

[7] Clark, K. Negation as failure. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, 1978, pp. 292–322.

[8] Collet, C., Huhns, M. N., and Shen, W.-M. Resource integration using a large knowledge base in Carnot. *IEEE Computer 24*, 12 (Dec 1991), 55–63.

[9] Dobbie, G., and Topor, R. On the declarative and procedural semantics of deductive object-oriented systems. *Journal of Intelligent Information Systems 4* (1995), 193–219.

[10] Eshghi, K., and Kowalski, R. A. Abduction compared with negation by failure. In *Proc. 6th International Conference on Logic Programming* (Lisbon, 1989), pp. 234–255.

[11] Faquhar, A., Dappert, A., Fikes, R., and Pratt, W. Integrating information sources using context logic. In *AAAI-95 Spring Symposium on Information Gathering from Distributed Heterogeneous Environments* (1995). To appear.

[12] Goh, C. H., Bressan, S., Madnick, S. E., and Siegel, M. D. Context interchange: Representing and reasoning about data semantics in heterogeneous systems. Sloan School Working Paper #3928, Sloan School of Management, MIT, 50 Memorial Drive, Cambridge MA 02139, Oct 1996.

[13] Goh, C. H., Madnick, S. E., and Siegel, M. D. Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, Nov 29–Dec 1 1994), pp. 337–346.

[14] Gruber, T. R. 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, 1991), J. A. Allen, R. Files, and E. Sandewall, Eds., Morgan Kaufmann, pp. 601–602.

[15] Imielinski, T. Intelligent query answering in rule based systems. *Journal of Logic Programming 4*, 3 (Sep 1987), 229–257.

[16] Jonker, W., and Schütz, H. The ECRC multidatabase system. In *Proc ACM SIGMOD* (1995), p. 490.

[17] Kakas, A. C., Kowalski, R. A., and Toni, F. Abductive logic programming. *Journal of Logic and Computation 2*, 6 (1993), 719–770.

[18] Kakas, A. C., and Mancarella, P. Database updates through abduction. In *Proceedings 16th International Conference on Very Large Databases* (Brisbane, Australia, 1990), pp. 650–661.

[19] Kifer, M., Lausen, G., and Wu, J. Logical foundations of object-oriented and frame-based languages. *JACM 4* (1995), 741–843.

[20] Kuhn, E., and Ludwig, T. VIP-MDBMS: A logic multidatabase system. In *Proc Int'l Symp. on Databases in Parallel and Distributed Systems* (1988).

[21] Landers, T., and Rosenberg, R. An overview of Multibase. In *Proceedings 2nd International Symposium for Distributed Databases* (1982), pp. 153–183.

[22] Lenat, D. B., and Guha, R. *Building large knowledge-based systems: representation and inference in the Cyc project.* Addison-Wesley Publishing Co., Inc., 1989.

[23] Litwin, W. O*SQL: A language for object oriented multidatabase interoperability. In *Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)* (Lorne, Victoria, Australis, Nov 1992), D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, Eds., North-Holland, pp. 119–138.

[24] LITWIN, W., AND ABDELLATIF, A. An overview of the multi-database manipulation language MDSL. *Proceedings of the IEEE 75*, 5 (1987), 621–632.

[25] LLOYD, J. W. *Foundations of logic programming*, 2nd, extended ed. Springer-Verlag, 1987.

[26] MCCARTHY, J. Generality in artificial intelligence. *Communications of the ACM 30*, 12 (1987), 1030–1035.

[27] MUMICK, I. S., AND PIRAHESH, H. Implementation of magic-sets in a relational database system. In *Proc. ACM SIGMOD* (Minneapolis, MN, May 1994), pp. 103–114.

[28] PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., AND WIDOM, J. Object exchange across heterogeneous information sources. In *Proc IEEE International Conference on Data Engineering* (March 1995).

[29] QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., AND WIDOM, J. Querying semistructured heterogeneous information. In *Proc International Conference on Deductive and Object-Oriented Databases* (1995).

[30] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. Context interchange using meta-attributes. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 377–386.

[31] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems 19*, 2 (June 1994), 254–290.

[32] SESHADRI, P., HELLERSTEIN, J. M., PIRAHESH, H., LEUNG, T. C., RAMAKRISHNAN, R., SRIVASTAVA, D., STUCKEY, P. J., AND SUDARSHAN, S. Cost-based optimization for magic: algebra and implementation. In *Proc. ACM SIGMOD* (Montreal, Quebec, Canada, June 1996), pp. 435–446.

[33] SHETH, A. P., AND LARSON, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys 22*, 3 (1990), 183–236.

[34] SIEGEL, M., AND MADNICK, S. A metadata approach to solving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases* (1991), pp. 133–145.

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

[36] TOMASIC, A., RASCHID, L., AND VALDURIEZ, P. Scaling heterogeneous databases and the design of DISCO. In *Proceedings of the 16th International Conference on Distributed Computing Systems* (Hong Kong, 1995).

[37] WIEDERHOLD, G. Mediators in the architecture of future information systems. *IEEE Computer 25*, 3 (March 1992), 38–49.