Context Knowledge Representation and Reasoning in the Context Interchange System

Stephane Bressan[1], Cheng Goh[2], Natalia Levina, Stuart Madnick, Ahmed Shah, Michael Siegel

Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

The Context Interchange Project presents a unique approach to the problem of semantic conflict resolution among multiple heterogeneous data sources. The system presents a semantically meaningful view of the data to the receivers (e.g. user applications) for all the available data sources. The semantic conflicts are automatically detected and reconciled by a Context Mediator using the context knowledge associated with both the data sources and the data receivers. The results are collated and presented in the receiver context. The current implementation of the system provides access to flat files, classical relational databases, on-line databases, and web services. An example application, using actual financial information sources, is described along with a detailed description of the operation of the system for an example query.

Keywords: context, integration, mediation, semantic heterogeneity, web wrapping

1. Introduction

In recent years the amount of information available has grown exponentially. While the availability of so much information has helped people become self-sufficient and get access to all the information handily, this has created another dilemma. All these data sources and the technologies that are employed by the data source providers do not provide sufficient logical connectivity (the ability to exchange data meaningfully). Logical connectivity is crucial because users of these sources expect each system to understand requests stated in their own terms, using their own concepts of how the world is defined and structured. As a result, any data integration effort must be capable of reconciling semantic conflicts among sources and receivers. This problem is generally referred to as the need for *semantic interoperability* among distributed data sources.

The Context Interchange Project at MIT [1,2] is studying the semantic integration of disparate information sources. Like other information integration projects (the SIMS project at ISI [3], the TSIMMIS project at Stanford [4], the DISCO project at Bull-INRIA [5], the Information Manifold project at AT&T [6], the Garlic project at IBM [7], the Infomaster project at Stanford [8]), we have adopted a Mediation architecture as outlined in Wiederhold's seminal paper [9].

In section 2, we present a motivational scenario of a user trying to access information from various actual data sources and the problems faced. Section 3 describes the current implementation of the Context mediation system. Section 4 presents a detailed discussion of the various subsystems, highlighting the context knowledge representation and reasoning, using the scenario outlined in section 2. Section 5 concludes our discussion.

2. Why Context Mediation ? – An Example Scenario

Consider an example of a financial analyst doing research on Daimler Benz. She needs to find out the net income, net sales, and total assets of Daimler Benz Corporation for the year ending 1993. In addition to that, she needs to know the closing stock price of Daimler Benz. She normally uses the financial data stored in the *Worldscope*[3] database. She recalls Jill, her co-worker telling her about two other databases, *Datastream*[4] and *Disclosure*[5] and how they contained much of the information that Jill needed. She starts off with *Worldscope* database. She knows that *Worldscope* has total assets for all the companies. She brings up a query tool and issues a query:

[1] Now at the National University of Singapore.
[2] We dedicate this work to the memory of Cheng Hian Goh (1965-1999).
[3] The *Worldscope* database is an extract from the Worldscope financial data source
[4] The *Datastream* database is an extract from the Datastream financial data source.

```
select company_name, total_assets from worldscope
where company_name = "DAIMLER-BENZ AG";
```

She immediately gets back the result:

DAIMLER-BENZ AG 5659478

Satisfied, she moves on and figures out after looking at the data information for the new databases that she can get the data on net income from *Disclosure* and net sales from *Datastream*. For net income, she issues the query:

```
select company_name, net_income from disclosure
where company_name = "DAIMLER-BENZ AG";
```

The query does not return any records. Puzzled, she checks for typos and tries again. She knows that the information exists. She tries one more time, this time entering a partial name for DAIMLER BENZ.

```
select company_name, net_income from disclosure
where company_name like "DAIMLER%";
```

She gets the record back:

DAIMLER BENZ CORP 615000000

She now realizes that the data sources do not conform to the same standards, as it becomes obvious from the names. Cautious, she presses on and issues the third query:

```
select name, total_sales from datastream
where  name like "DAIMLER%";
```

She gets the result:

DAIMLER-BENZ 9773092

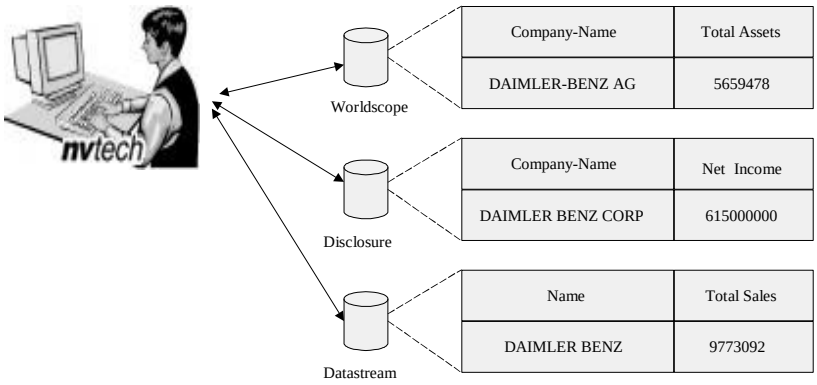

Figure 1: Integrating Information from Multiple Sources

As she is putting the results together, she realizes that there are a number of things unusual about the data set shown in Figure 1. First of all, the Total Sales are twice as much as the total assets of the company, which is highly unlikely for a company like Daimler Benz. What is even more disturbing is that net income is more than 60 times as much as total sales. She immediately realizes something is wrong and grudgingly opens up the documents that came with the databases. Upon studying the documentation, she finds out some interesting facts about the data that she was using so gaily. She finds out that *Datastream* has a scale factor of 1000 for all the financial amounts, while *Disclosure* uses a scale factor of 1. In addition, both *Disclosure* and *Datastream* use the country of incorporation to identify the currency, which, in the case of Daimler-Benz, would be German Deutschmarks. She knew that *Worldscope* used a scale factor of 1000 but at least everything was in U.S Dollars. Now she has to reconcile all the information by finding a data source (possibly on the web) that contains the historical currency exchange rates (i.e. as of end of the year 1993). In addition she still has to somehow find another data source to get the latest stock price for Daimler

[5] The *Disclosure* database, once again, is an extract from the original Disclosure financial data source. By coincidence, although all three sources were originally provided by independent companies, they are all currently owned by a single company, Primark.

Benz. For that, she knows she will first have to find out the ticker for Daimler Benz and then look up the price using one of the many stock quote servers on the web.

The Context Mediation system can be used to automatically detect and resolve all the semantic conflicts between all the data sources being used and can present the results to the user in the format that she is familiar with. In the above example, if the analyst were using the Context Mediation system instead, all she would have to do would be to formulate and ask a single query without having to worry about the underlying differences between the data. Both her request and the result would be formulated in her preferred context (e.g. *Worldscope*). The multi-source query, **Query1**, could be stated as follows:

```
select worldscope.total_assets, datastream.total_sales,
disclosure.net_income, quotes.Last
from worldscope, datastream, disclosure, quotes  where
worldscope.company_name = "DAIMLER-BENZ AG" and
datastream.as_of_date = "01/05/94"  and
worldscope.company_name = datastream.name and
worldscope.company_name = disclosure.company_name and
worldscope.company_name = quotes.cname ;
```

The system would then detect and reconcile the conflicts encountered by the analyst.

3. Overview of the COIN Project

The COntext INterchange (COIN) strategy seeks to address the problem of *semantic interoperability* by consolidating distributed data sources and providing a unified view. COIN technology presents all data sources as SQL relational databases by providing generic wrappers for them. The underlying integration strategy, called the COIN model, defines a novel approach for mediated [9] data access in which semantic conflicts among heterogeneous systems are automatically detected and reconciled by the *Context Mediator*.

3.1 The COIN Framework

The COIN framework is composed of both a data model and a logical language, COINL [11], derived from the family of F-Logic [10]. The data model and language are used to define the *domain model* of the receiver and data source and the *context* [12] associated with them. The data model contains the definitions for the "types" of information units (called *semantic types*) that constitute a common vocabulary for capturing the semantics of data in disparate systems. *Contexts*, associated with both information sources and receivers, are collections of statements defining how data should be interpreted and how potential conflicts (differences in the interpretation) should be resolved. Concepts such as *semantic-objects*, *attributes*, *modifiers*, and *conversion functions* define the semantics of data inside and across *contexts*. Together with the deductive and object-oriented features inherited from F-Logic, the COIN data model and COINL constitute an appropriate and expressive framework for representing semantic knowledge and reasoning about semantic heterogeneity.

3.2 Context Mediator

The *Context Mediator* is the heart of the COIN project. Mediation is the process of rewriting queries posed in the receiver's *context* into a set of mediated queries where all actual conflicts are explicitly resolved and the result is reformulated in the receiver context. This process is based in an abduction [13] procedure that determines what information is needed to answer the query and how conflicts should be resolved by using the axioms in the different *contexts* involved. Answers generated by the mediation unit can be both extensional and intentional. Extensional answers correspond to the actual data retrieved from the various sources involved. Intentional answers, on the other hand, provide only a characterization of the extensional answer without actually retrieving data from the data sources. In addition, the mediation process supports queries on the semantics of data that are implicit in the different systems. There are referred to as *knowledge-level queries* as opposed to *data-level queries* that are enquires on the factual data present in the data sources. Finally, integrity knowledge on one source or across sources can be naturally involved in the mediation process to improve the quality and information content of the mediated queries and ultimately aid in the optimization of the data access.

3.3 System Perspective

From a system perspective, the COIN strategy combines the best features of the *loose-* and *tight-coupling* approaches to *semantic interoperability* [14] among autonomous and heterogeneous systems. Its modular design and

implementation, depicted in Figure 2, funnels the complexity of the system into manageable chunks, enables sources and receivers to remain loosely-coupled to one another, and sustains an infrastructure for data integration. This modularity, both in the components and the protocol, also keeps our infrastructure scalable, extensible, and accessible [2]. By *scalability*, we mean that the complexity of creating and administering the mediation services does not increase exponentially with the number of sources and receivers that participate. *Extensibility* refers to the

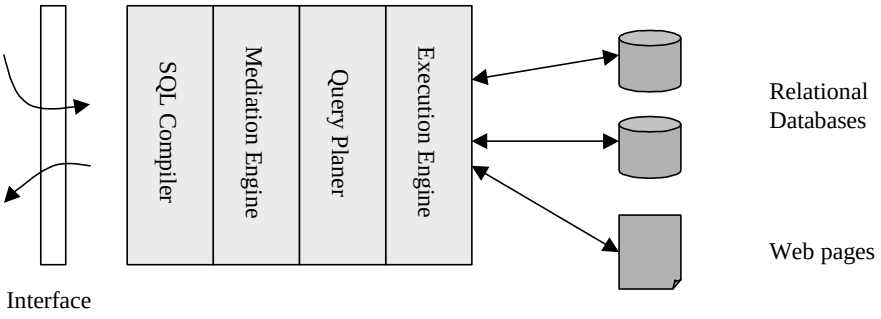

Figure 2: Context Mediator

ability to incorporate changes into the system in a graceful manner; in particular, local changes do not have adverse effects on other parts of the system. Finally, *accessibility* refers to how a user, in terms of its ease-of-use, perceives the system and flexibility in supporting a variety of queries.

3.4 Application Domains

The COIN technology can be applied to a variety of scenarios where information needs to be shared amongst heterogeneous sources and receivers. The need for this novel technology in the integration of disparate data sources can be readily seen in many examples.

We have already seen one application of context mediation technology in the financial domain in the previous section. There are many information providers that provide historical data and other research both to institutions (investment banks, brokerages) as well as individual investors. Most of the time this information is presented in different formats and must be interpreted with different rules. Obvious examples are scale-factors and currency of monetary figures. Much more subtle mismatches of assumptions across sources or even inside one source can be critical in the process of financial decision making. Many such examples have been discovered as part of this research effort.

In the domain of manufacturing inventory control, the ability to access design, engineering, manufacturing and inventory data pertaining to all parts, components, and assemblies vital to any large manufacturing process. Typically, thousands of contractors play roles and each contractor tends to set up its data in its own individualistic manner. Managers may need to reconcile inputs received from various contractors in order to optimize inventory levels and ensure overall productivity and effectiveness. As another example, the modern health care enterprise lies at the nexus of several different industries and institutions. Within a single hospital, different departments (e.g. internal medicine, medical records, pharmacy, admitting, and billing) maintain separate information systems yet must share data in order to ensure high levels of care. Medical centers and local clinics not only collaborate with one another but with State and Federal regulators, insurance companies, and other payer institutions. This sharing requires reconciling differences such as those of procedure codes, medical supplies, classification schemes, and patient records. Similar situations have been found in almost every industry. Other industries studied in this research effort include government and military organizations.

4. The COIN Architecture

The feasibility and features of this proposed strategy have been demonstrated in a working system that provides mediated access to both on-line structured databases and semi-structured data sources such as web sites. The infrastructure leverages on the World Wide Web in a number of ways. First, COIN relies on the hypertext transfer protocol for the physical connectivity among sources and receivers and the different mediation components and services. Second, COIN employs the hypertext markup Language and Java for the development of portable user interfaces. Figure 3 shows the architecture of the COIN system. It consists of three distinct groups of processes.

- **Client Processes** provide the interaction with receivers and route all database requests to the Context Mediator. An example of a client process is the *multi-database browser* [15], which provides a point-and-click interface for formulating queries to multiple sources and for displaying the answers obtained. Specifically, any application program that issues queries to one or more sources can be considered a client process.
- **Server Processes** refer to *database gateways* and *wrappers*. Database gateways provide physical connectivity to a database on a network. The goal is to insulate the Mediator Process from the idiosyncrasies of different database management systems by providing a uniform protocol for database access as well as canonical query language (and data model) for formulating the queries. Wrappers provide richer functionality by allowing semi-structured documents on the World Wide Web to be queried as if they were relational databases. This is accomplished by defining an *export schema* for each of these web sites and describing how attribute-values can be extracted from a web site using a finite automaton with pattern matching [16].
- **Mediator Processes** refer to the system components that collectively provide the mediation services. These include SQL-to-datalog compiler, context mediator, and query planner/optimizer and multi-database executioner. SQL-to-Datalog compiler translates a SQL query into its corresponding datalog format. The Context Mediator rewrites the user-provided query into a mediated query with all the conflicts resolved. The planner/optimizer produces a query evaluation plan based on the mediated query. The multi-database executioner processes the query plan generated by the planner. It dispatches sub-queries to the server processes, collates the intermediary results, converts the result into the client context, and returns the reformulated answer to the client processes.

Of these three distinct groups of processes, the most relevant to our discussion of context knowledge and reasoning are the mediator processes. We will start by explaining the domain model and then discuss the prototype system.

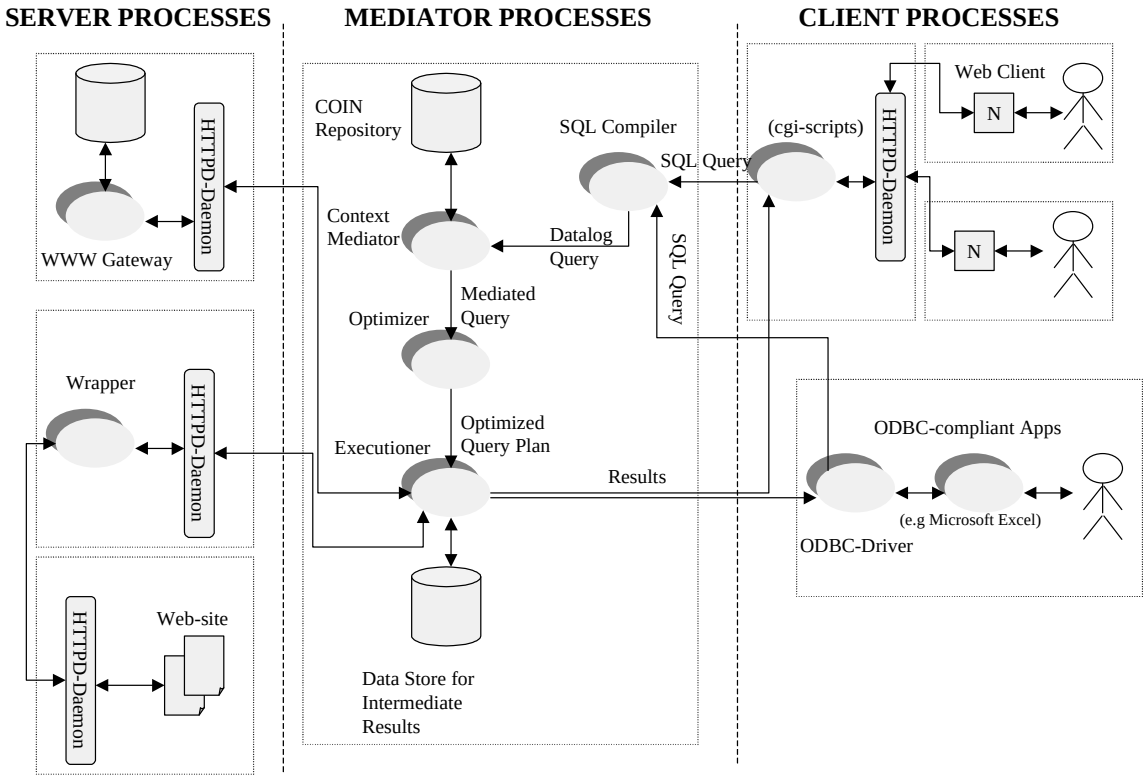

Figure 3: COIN System Overview

4.1 Domain Model and Context definition

The first thing that we need to do is specify the domain model for the domain that we are working in. A *domain model* specifies the semantics of the "types'" of information units, which constitutes a common vocabulary used in capturing the semantics of data in disparate sources. In other words it defines the ontology which will be used. The various semantic types, the type hierarchy, and the type signatures (for attributes and modifiers) are all defined in

the domain model. Types in the generalized hierarchy are rooted to system types, i.e. types native to the underlying system such as integers, strings, real numbers etc.

Figure 4 depicts part of the domain model that is used in our example. In the domain model described, there are three kinds of relationships expressed.

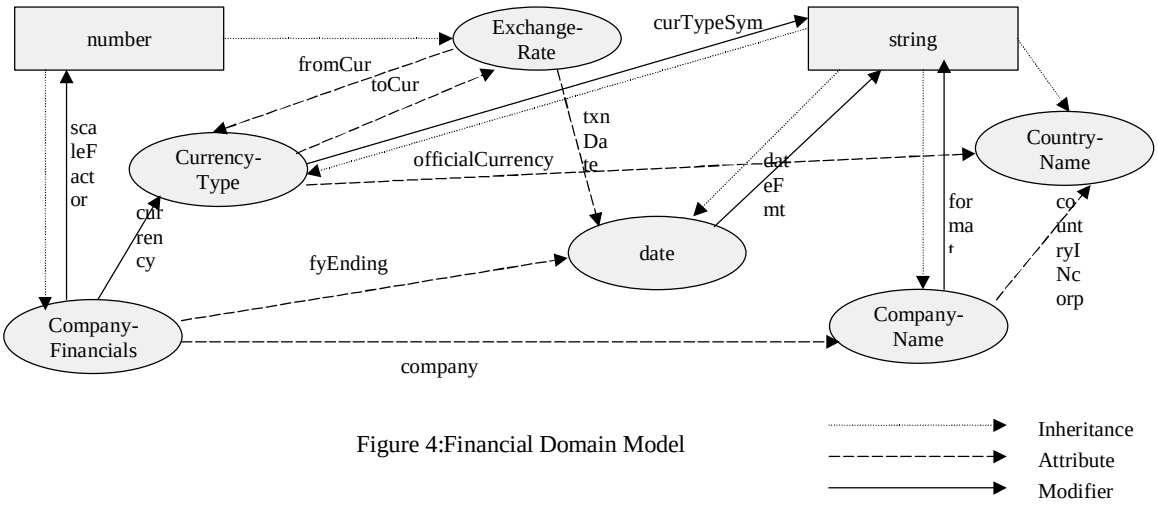

Figure 4:Financial Domain Model

- **Inheritance:** This is the classic type inheritance relationship. All semantic types inherit from basic system types. In the domain model, type `companyFinancials` inherits from basic type string.
- **Attributes:** In COIN [17], objects have two forms of properties, those which are structural properties of the underlying data source and those that encapsulate the underlying assumptions about a particular piece of data. *Attributes* access structural properties of the semantic object in question. For instance, the semantic type `companyfinancials` has two attributes, `company` and `fyEnding`. Intuitively, these attributes define a relationship between objects of the corresponding semantic types. Here, the relationship formed by the `company` attribute states that for any company financial in question, there must be corresponding company to which that company financial belongs. Similarly, the `fyEnding` attribute states that every company financial object has a date when it was recorded.
- **Modifiers:** These define a relationship between semantic objects of the corresponding semantic types. The difference though is that the values of the semantic objects defined by the modifiers have varying interpretations depending on the context. Looking once again at the domain model, the semantic type `companyFinancials` defines two modifiers, `scaleFactor` and `currency`. The value of the object returned by the modifier `scaleFactor` depends on a given context.

Once we have defined the domain model, we need to define the contexts for all the sources. In our case, we have several data sources with the assumptions about their data in figure 5.

A simplified view of what the context might be for the *Worldscope* data source is:

```
modifier(companyFinancials, O, scaleFactor, c_ws, M):-
       cste(basic, M, c_ws, 1000).
modifier(companyFinancials, O, currency, c_ws, M):-
       cste(currencyType, M, c_ws, "USD").
modifier(date, O, dateFmt, c_ws, M):-
       cste(basic, M, c_ws, "American Style /").
```

Datasource	Scale Factor	Currency	Date Format
Worldscope	1000	USD	American "/"
Disclosure	1	Local	American "/"
datastream	1000	Local	European "-"
Olsen	1	Local	European "/"
Quote	1	USD	American "/"

Figure 5: Context Table

Each statement refers to a potential conflict that needs to be resolved by the system. Yet another way to look at it is that each statement corresponds to a modifier relation in the actual domain model. From the domain model shown in Figure 4, we notice that the object *CompanyFinancials* has two modifiers, *scaleFactor* and *currency*. Correspondingly, the first two statements define these two modifiers. Looking at the context table in Figure 5, we notice that the value of the *scaleFactor* in the *Worldscope* context is 1000. The first statement represents that fact. It states that the modifier *scaleFactor* for the object *O* of type *companyFinancials* in the context *c_ws* is the object *M* where (the second line) the object *M* is a constant (*cste*) of type *basic* and has a value of 1000 in the context *c_ws*. In the case of the *Worldscope* data source, all the financial amounts have a scale factor of 1000. That means that in order to get the actual amount of total assets, we will have to multiply the amount returned from the data source by 1000. The next clause determines the *currency* to be in *USD* (i.e., US dollars). The last clause tells the system that the format of the date string in the *Worldscope* is of type American Style with "/" as the delimiting character (mm/dd/yy).

One last thing that needs to be provided as part of a context is the set of conversion functions between different contexts. An example is the conversion between scale factors in different contexts. Following is the conversion routine that is used when scale factors are not equal. The function states that in order to perform conversion of the modifier *scaleFactor* for the object *_O* of semantic type *companyFinancials* in the context *Ctxt* where the modifier value in the source is *Mvs* and the object *_O's* value in the source context is *Vs* and the modifier value in the target context is *Mvt* and the object *_O's* value in the target context is *Vt*, we first find out the *Ratio* between the modifier value in the source context and the modifier value in the target context. We then determine the object's value in the target context by multiplying its value in the source context with the *Ratio*. *Vt* now contains the appropriately scaled value for the object *_O* in the target context. Note that these conversion rules are defined independent of any specific source or receiver context, the Context Mediator determines if or when such a conversion is needed.

```
cvt(companyFinancials, _O, scaleFactor, Ctxt,
  Mvs, Vs, Mvt, Vt) :-
       Ratio is Mvs / Mvt,
       Vt is Vs * Ratio.
```

4.2 Elevation Axioms

The mapping of data and data-relationships from the sources to the domain model is accomplished via the elevation axioms. There are three distinct operations that define the elevation axioms:

- Define a virtual semantic relation corresponding to each extensional relation.
- Assign to each semantic object defined its value in the context of the source.
- Map the semantic objects in the semantic relation to semantic types defined in the domain model and make explicit any implicit links (attribute initialization) represented by the semantic relation.

We will use the example of the relation Worldscope to show how the relation is elevated. The Worldscope relation is a table in an Oracle database and has the following columns:

Name	Type
COMPANY_NAME	VARCHAR2(80)

LATEST_ANNUAL_FINANCIAL_DATE	VARCHAR2(10)
CURRENT_OUTSTANDING_SHARES	NUMBER
NET_INCOME	NUMBER
SALES	NUMBER
COUNTRY_OF_INCORP	VARCHAR2(40)
TOTAL_ASSETS	NUMBER

And here is what part of the elevated relation looks like:

```
'WorldcAF_p'(
       skolem(companyName, Name, c_ws, 1, 'WorldcAF'( Name, FYEnd, Shares, Income,
Sales, Assets, Incorp)),
       skolem(date, FYEnd, c_ws, 2, 'WorldcAF'( Name, FYEnd, Shares, Income, Sales,
Assets, Incorp)),
       skolem(basic, Shares, c_ws, 3, 'WorldcAF'( Name, FYEnd, Shares, Income, Sales,
Assets, Incorp)),
       skolem(companyFinancials, Income, c_ws, 4, 'WorldcAF'( Name, FYEnd, Shares,
Income, Sales, Assets, Incorp)),
       skolem(companyFinancials, Sales, c_ws, 5, 'WorldcAF'( Name, FYEnd, Shares,
Income, Sales, Assets, Incorp)),
       skolem(companyFinancials, Assets, c_ws, 6, 'WorldcAF'( Name, FYEnd, Shares,
Income, Sales, Assets, Incorp)),
       skolem(countryName, Incorp, c_ws, 7, 'WorldcAF'( Name, FYEnd, Shares, Income,
Sales, Assets, Incorp))
    )   :- 'WorldcAF'(Name, FYEnd, Shares, Income, Sales, Assets, Incorp).
```

We first define a semantic relation for *Worldscope*. A semantic relation is then defined on the semantic objects in the corresponding relation attributes. The data elements derived from the extensional relation are mapped to semantic objects. These semantic objects define a unique object-id for each data element. In the example above each skolem term defines a unique semantic object corresponding to each attribute of the extensional relation. In addition to mapping each physical relation to a corresponding semantic object, we also define and initialize other relations defined in the domain model. The relations that come under this category are attribute and modifiers.

4.3 Mediation System

In the following sections, we will describe each subsystem. We will use the application scenario of the financial analyst trying to gather information about Daimler Benz Corporation. We will use **Query1**, as presented in Section 2.1, as an example multi-source query. We then describe the application as it is programmed, explaining the domain and how the context information for various sources is specified. Then we will follow the query as it passes through each subsystem.

Query1 is intended to gather financial data for the Daimler Benz Corporation for the year 1993. We get net assets from the Worldscope data source, net sales from the Datastream data source, net income from the Disclosure data source and the latest quotes from Quote data source, which happens to be the CNN web quote server. We will be asking the query in the Worldscope context (i.e., the result of the query will be returned in the Worldscope context.)

4.3.1 SQL to Datalog Query Compiler

The first step is to parse the SQL into its corresponding datalog form and using the elevation axioms it elevates the data sources into its corresponding elevated data objects. The corresponding datalog for the SQL query above is:

```
answer(total_assets, total_sales, net_income, last) :-
        WorldcAF_p(V27, V26, V25, V24, V23, V22, V21),
        DiscAF_p(V20, V19, V18, V17, V16, V15, V14),
        DStreamAF_p(V13, V12, V11, V10, V9, V8),
        quotes_p(V7, q_last),
        Value(V27, c_ws, V5),
        V5 = "DAIMLER-BENZ AG",
        Value(V13, c_ws, V4),
        V4 = "01/05/94",
        Value(V12, c_ws, V3),
        V5 = V3,
```

```
Value(V20, c_ws, V2),
V5 = V2,
Value(V7, c_ws, V1),
V5 = V1,
Value(V22, c_ws, total_assets),
Value(V17, c_ws, total_sales),
Value(V11, c_ws, net_income),
Value(q_last, c_ws, last).
```

As can be seen, the query now contains elevated data sources along with a set of predicates that map each attribute to its value in the corresponding context. Since the user asked the query in the Worldscope context (denoted by *c_ws*), the last four predicates in the translated query ascertain that the actual values returned as the solution of the query need to be in the Worldscope context. The resulting unmediated datalog query is then fed to the mediation engine.

4.3.2 Mediation Engine

The mediation engine is the part of the system that detects and resolves possible semantic conflicts. In essence, the mediation is a query rewriting process. The actual mechanism of mediation is based on an Abduction Engine [13]. The engine takes a datalog query and a set of domain model axioms and computes a set of abducted queries such that the abducted queries have all the differences resolved. The system does that by incrementally testing for potential semantic conflicts and introducing conversion functions for the resolution of those conflicts. The mediation engine as its output produces a set of queries that take into account all the possible cases given the various conflicts. Using the above example and with the domain model and contexts stated above, we would get the set of abducted queries shown below:

```
answer(V108, V107, V106, V105) :-
        datexform(V104, "European Style -", "01/05/94", "American Style /"),
        Name_map_Dt_Ws(V103, "DAIMLER-BENZ AG"),
        Name_map_Ds_Ws(V102, "DAIMLER-BENZ AG"),
        Ticker_Lookup2("DAIMLER-BENZ AG", V101, V100),
        WorldcAF("DAIMLER-BENZ AG", V99, V98, V97, V96, V108, V95),
        DiscAF(V102, V94, V93, V92, V91, V90, V89),
        V107 is V92 * 0.001,
        Currencytypes(V89, USD),
        DStreamAF(V104, V103, V106, V88, V87, V86),
        Currency_map(USD, V86),
        quotes(V101, V105).

answer(V85, V84, V83, V82) :-
        datexform(V81, "European Style -", "01/05/94", "American Style /"),
        Name_map_Dt_Ws(V80, "DAIMLER-BENZ AG"),
        Name_map_Ds_Ws(V79, "DAIMLER-BENZ AG"),
        Ticker_Lookup2("DAIMLER-BENZ AG", V78, V77),
        WorldcAF("DAIMLER-BENZ AG", V76, V75, V74, V73, V85, V72),
        DiscAF(V79, V71, V70, V69, V68, V67, V66),
        V84 is V69 * 0.001,
        Currencytypes(V66, USD),
        DStreamAF(V81, V80, V65, V64, V63, V62),
        Currency_map(V61, V62),
        <>(V61, USD),
        datexform(V60, "European Style /", "01/05/94", "American Style /"),
        olsen(V61, USD, V59, V60),
        V83 is V65 * V59,
        quotes(V78, V82).

answer(V58, V57, V56, V55) :-
        datexform(V54, "European Style -", "01/05/94", "American Style /"),
        Name_map_Dt_Ws(V53, "DAIMLER-BENZ AG"),
        Name_map_Ds_Ws(V52, "DAIMLER-BENZ AG"),
        Ticker_Lookup2("DAIMLER-BENZ AG", V51, V50),
        WorldcAF("DAIMLER-BENZ AG", V49, V48, V47, V46, V58, V45),
        DiscAF(V52, V44, V43, V42, V41, V40, V39),
        V38 is V42 * 0.001,
        Currencytypes(V39, V37),
        <>(V37, USD),
        datexform(V36, "European Style /", V44, "American Style /"),
        olsen(V37, USD, V35, V36),
        V57 is V38 * V35,
```

```
        DStreamAF(V54, V53, V56, V34, V33, V32),
        Currency_map(USD, V32),
        quotes(V51, V55).

answer(V31, V30, V29, V28) :-
        datexform(V27, "European Style -", "01/05/94", "American Style /"),
        Name_map_Dt_Ws(V26, "DAIMLER-BENZ AG"),
        Name_map_Ds_Ws(V25, "DAIMLER-BENZ AG"),
        Ticker_Lookup2("DAIMLER-BENZ AG", V24, V23),
        WorldcAF("DAIMLER-BENZ AG", V22, V21, V20, V19, V31, V18),
        DiscAF(V25, V17, V16, V15, V14, V13, V12),
        V11 is V15 * 0.001,
        Currencytypes(V12, V10),
        <>(V10, USD),
        datexform(V9, "European Style /", V17, "American Style /"),
        olsen(V10, USD, V8, V9),
        V30 is V11 * V8,
        DStreamAF(V27, V26, V7, V6, V5, V4),
        Currency_map(V3, V4),
        <>(V3, USD),
        datexform(V2, "European Style /", "01/05/94", "American Style /"),
        olsen(V3, USD, V1, V2),
        V29 is V7 * V1,
        quotes(V24, V28).
```

The mediated query contains four sub-queries. Each of the sub-queries accounts for a potential semantic conflict. For example, the first sub-query deals with the case when there is no currency conversion conflict (i.e., source and receiver use same currency). While the second sub-query takes into account the possibility of currency conversion. Resolving the conflicts may sometime require introducing intermediate data sources. Figure 5 listed some of the context differences in the various data sources that we use for our example. Looking at the table, we observe that one of the possible conflicts is different data sources using different currencies. In order to resolve that difference, the mediation engine has to introduce an intermediary data source. The source used for this purpose is a currency conversion web site (*http://www.oanda.com*) and is referred to as *olsen*. In order to resolve the currency conflict in the second sub-query, the *olsen* source is used to convert the currency to correctly represent data in the currency specified as of the specified date in the specified context. Note that it is the mediator, using the context knowledge, that determines that currency conversion was needed in this case.

4.3.3 Query planner and optimizer

The query planner module takes the set of datalog queries produced by the mediation engine and produces a query plan. It ensures that an executable plan exists which will produce a result that satisfies the initial query. This is necessitated by the fact that there are some sources that impose restrictions on the type of queries that they can service. In particular, some sources may require that some of the attributes must always be bounded while making queries to those sources. Another limitation that sources might have is the kinds of operators that they can handle. One example is that most web sources do not provide an interface that supports all the SQL operators, or they might require that some attributes in queries be always bound. Once the planner ensures than an executable plan exists, it generates a set of constraints on the order in which the different sub-queries can be executed. Under these constraints, the optimizer applies standard optimization heuristics to generate the query execution plan. The query execution plan is essentially an algebraic operator tree in which each operation is represented by a node. There are two types of nodes:

- **Access Nodes:** Access nodes represent access to remote data sources. Two subtypes of access nodes are:
  - *sfw Nodes:* These nodes represent access to data-sources that do not require input bindings from other sources in the query.
  - *join-sfw Node:* These node have a dependency in that they require input from other data sources in the query. Thus these nodes have to come after the nodes that they depend on while traversing the query plan tree.
- **Local Nodes:** These nodes represent local operations in local execution engine. There are four subtypes of local nodes.
  - *Join Node:* Joins two trees
  - *Select Node:* This node is used to apply conditions to intermediate results.
  - *CVT Node:* This node is used to apply conversion functions to intermediate query result.
  - *Union Node:* This node represents a union of the results obtained by executing the sub-nodes.

Each node carries additional information about what data-source to access (if it is an access node) and other information that is used by the runtime engine. Some of the information that is carried in each node is a list of attributes in the source and their relative position, list of condition operations and any literals and other information like the conversion formula in the case of a conversion node. The query plan for the first sub-query of the mediated query is shown in the Appendix. The query plan that is produced by the planner is then forwarded to the runtime engine.

4.3.4 Runtime engine

The runtime execution engine executes the query plan. Given a query plan, the execution engine traverses the query plan tree in a depth-first manner starting from the root node. At each node, it computes the sub-trees for that node and then applies the operation specified for that node. For each sub-tree the engine recursively descends down the tree until it encounters an access node. For that access node, it composes a SQL query and sends it off to the remote source. The results of that query are then stored in the local store. Once all the sub-trees have been executed and all the results are in the local store, the operation associated with that node is executed and the results collected. This operation continues until the root of the query is reach. At this point the execution engine has the required set of results corresponding to the original query. These results are then sent back to the user and the process is completed.

4.4 Web Wrapper

The original query used in our example, contained access to a quote server to get the most recent quotes for the company in question, i.e. Daimler-Benz. As opposed to the rest of the sources, the quote server that we used is a web quote server. In order to access the web sources such as this one, we have developed a technology that lets users treat web sites as relational data sources. Users can then issue SQL queries just as they would to any relation in the relational domain, thus combining multiple sources and creating queries as the one above. This technology is called *web-wrapping* and we have an implementation for this technology which is called *web wrapping engine* [18]. Using the web wrapper engine (web wrapper for short) the application developers can very rapidly *wrap* a structured or semi-structured web site and export the schema for the users to query against. Once the source has been wrapped, it can be used as a relational source in any query.

4.4.1 Web wrapper architecture

Figure 6 shows the architecture of the web wrapper. The system takes the SQL query as input. It parses the query along with the specifications for the given web site. A query plan is then constituted. The plan constitutes of what web sites to send http requests and what documents on those web sites. The executioner then executes the plan. Once the pages are fetched, the executioner then extracts the required information from the pages and presents that to the user.

4.4.2 Wrapping a web site

For our query, the relation *quote* is actually a web quote server that we access using our web wrapper. In order to *wrap* a site, you need to create a specification file. For each Web page or set of Web pages the generic Web Wrapper engine utilizes a specification file to guide it through the data extractions process. The specification file contains information about the locations on the web for both input and output data. The Web Wrapper Engine utilizes this information during query execution to get information back from the set of Web pages as if it were a relational database. This file is plain text file and contains information such as the exported schema, the URL of the web site to access, and a regular expression that will be used to extract the actual information from the web page. In our example we use the *CNN* quote server to get quotes.

A simplified specification file is included below:

```
#HEADER
#RELATION=quotes
#HREF=GET http://qs.cnnfn.com
#EXPORT= quotes.Cname quotes.Last
#ENDHEADER

#BODY
#PAGE
#HREF = POST http://qs.cnnfn.com/cgibin/stockquote?
  symbols=##quotes.Cname##
```

```
#CONTENT=last:  </font><FONT
  SIZE=+1><b>##quotes.Last##</FONT></TD>
#ENDPAGE
#ENDBODY
```

The specification has two parts, *Header* and *Body*. The *Header* part specifies information about the name of the relation and the exported schema. In the above case, the schema that we decided to export has two attributes, *Cname*, the ticket of the company and *Last* the latest quote. The *Body* Portion of the file specifies how to actually access the page (as defined in the *HREF* field) and what regular expression to use (as defined in the *CONTENT* field). Once the specification file is written and placed where the web wrapper can read it, we are ready to use the system. We can start making queries against the new relation that we just created.

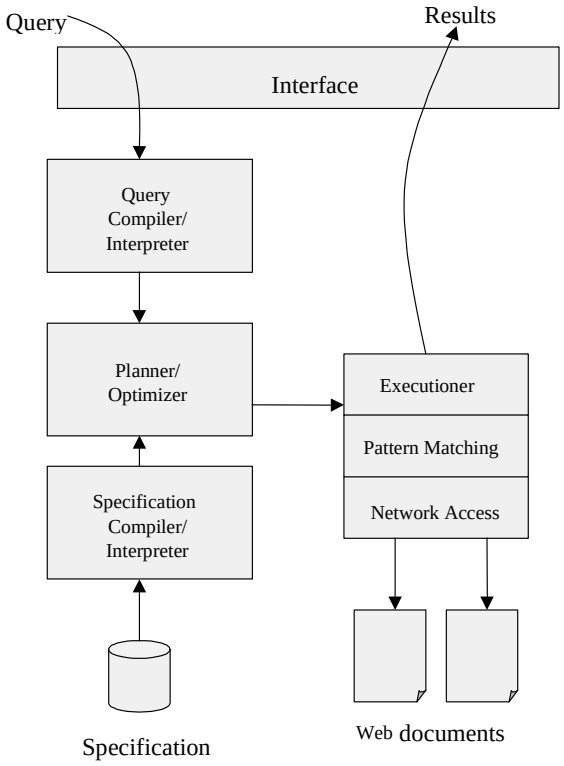

Figure 6: Web Wrapper Architecture

4.4.3 Web Wrapping and XML

The eXtensible Markup Language (XML) for Web pages is becoming increasingly accepted. This provides opportunities for the Web Wrapping Engine, in particular, and the Context Interchange System, in general. First, the XML tags provide a much easier and explicit demarcation of the location of fields within a web page. That makes the extraction of data much simpler for the Web Wrapper. On the other hand, XML is primarily a syntactic facility. You may have a tag <PRICE>, but XML does not provide the semantics, such as what is the currency of the price, does it include tax, does it include shipping costs, etc. The Context Interchange approach is the next step in the evolution of XML and the Web to provide the semantics that is so critical to the effective exchange of information.

5. Conclusions

In this paper, we have described a novel approach to the problem of resolving semantic differences between disparate information sources by automatically detecting and resolving semantic conflicts between those sources based on the knowledge of the contexts of those data sources in a particular domain. We have also described and explained the architecture and implementation of the prototype, and discussed the prototype at work by using an example scenario. More details pertaining to this scenario and a demonstration of its operation can be found at `http://context.mit.edu/~coin/demos/tasc/saved/queries/q11.html`.

Acknowledgements

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 Ph.D. 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 came 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...

Work reported herein has been supported, in part, by the Advanced Research Projects Agency (ARPA) and the USAF/Rome Laboratory under contract F30602-93-C-0160, TASC, Inc. (and its parent, PRIMARK Corporation), the MIT PROductivity From Information Technology (PROFIT) Program, and the MIT Total Data Quality Management (TDQM) Program. Information about the Context Interchange project can be obtained at `http://context.mit.edu/~coin`.

Bibliography

[1] Adil Daruwala, Cheng Goh, Scot Hofmeister, Karim Husein, Stuart Madnick, Michael Siegel. ``The Context Interchange Network Prototype", Center For Information System Laboratory (CISL), working paper, #95-01, Sloan School of Management, Massachusetts Institute of Technology, February 1995

[2] Goh, C., Madnick, S., Siegel, M. ``Context Interchange: Overcoming the challenges of Large-scale interoperable database systems in a dynamic environment". Proc. of Intl. Conf. On Information and Knowledge Management. 1994.

[3] Arens, Y. and Knobloch, C. ``Planning and reformulating queries for semantically-modeled multidatabase". Proc. of the Intl. Conf. on Information and Knowledge Management. 1992

[4] Garcia-Molina, H. ``The TSIMMIS Approach to Mediation: Data Models and Languages". Proc. of the Conf. on Next Generation Information Technologies and Systems. 1995.

[5] Tomasic, A. Rashid, L., and Valduriez, P. ``Scaling Heterogeneous databases and the design of DISCO". Proc. of the Intl. Conf. on Distributed Computing Systems. 1995.

[6] Levy, A., Srivastava, D. and Krik, T. ``Data Model and Query Evaluation in Global Information Systems". Journal of Intelligent Information Systems. 1995.

[7] Papakonstantinou, Y., Gupta, a., and Haas, L. ``Capabilities-Based Query Rewriting in Mediator Systems". Proc. of the 4th Intl. Conf. on Parallel and Distributed Information Systems. 1996.

[8] Duschka, O., and Genesereth, M. ``Query Planning in Infomaster". http://infomaster.standord.edu. 1997.

[9] Wiederhold, G. ``Mediation in the Architecture of Future Information Systems". Computer, 23(3), 1992.

[10] Kifer, M., Lausen, G., and Wu, J. Logical Foundations of Object-oriented and Frame-based Languages. JACM 5 (1995), pp. 741-843.

[11] Pena, F. PENNY: A Programming Language and Compiler for the Context Interchange Project. CISL Working Paper #97-06, 1997

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

[13] KaKas, A. C., Kowalski, R. A. and Toni, F. Abductive Logic Programming. Journal of Logic and Computation 2, 6 (1993), pp. 719-770.
[14] Sheth, A. P., and Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys 22, 3 (1990), pp. 183-236
[15] Jakobiasik, M. Programming the web-design and implementation of a multidatabase browser. Technical Report, CISL, WP #96-04, 1996
[16] Qu, J. F. Data wrapping on the world wide web. Technical Report, CISL WP #96-05, 1996
[17] Goh, C. H. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. PhD thesis, Sloan School of Management, Massachusetts Institute of Technology, 1996.
[18] Bressan, S., Bonnet, P. Extraction and Integration of Data from Semi-structured Documents into Business Applications. To be published.
[19] Madnick, S. Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Heterogeneity. Proceedings of the IEEE Meta-data Conference, April 1999.

APPENDIX: Part of the Query Execution Plan
(the entire plan can be found at http://context.mit.edu/~coin/demos/tasc/saved/saved-results/q11_t3.html)

```
SELECT
 ProjL: [att(1, 5), att(1, 4), att(1, 3), att(1, 2)]
 CondL: [att(1, 1) = "01/05/94"]
  JOIN-SFW-NODE DateXform
  ProjL: [att(1, 3), att(2, 4), att(2, 3), att(2, 2), att(2,1)]
  CondL: [att(1, 1) = att(2, 5), att(1, 2) = "European Style
  -", att(1, 4) = "American Style /"]
   CVT-NODE 'V18' is 'V17' * 0.001
   Projl: [att(2, 1), att(1, 1), att(2, 2), att(2, 3), att(2, 4)]
   Condl: ['V17' = att(2, 5)]
    JOIN-SFW-NODE quotes
    ProjL: [att(2, 1), att(2, 2), att(1, 2), att(2, 3), att(2, 4)]
    CondL: [att(1, 1) = att(2, 5)]

     JOIN-NODE
     ProjL: [att(2, 1), att(2, 2), att(2, 3), att(2, 4), att(2, 5)]
     CondL: [att(1, 1) = att(2, 6)]
      SELECT
      ProjL: [att(1, 2)]
      CondL: [att(1, 1) = 'USD']
       SFW-NODE Currency_map
       ProjL: [att(1, 1), att(1, 2)]
       CondL: []

      JOIN-NODE
      ProjL: [att(2, 1), att(1, 2), att(1, 3),att(2, 2),att(2, 3),att(1, 4)]
      CondL: [att(1, 1) = att(2, 4)]
       SFW-NODE Datastream
       ProjL: [att(1, 2), att(1, 3), att(1, 1), att(1, 6)]
       CondL: []

       JOIN-NODE
       ProjL: [att(2, 1), att(2, 2), att(2, 3), att(2, 4)]
       CondL: [att(1, 1) = att(2, 5)]
        SELECT
        ProjL: [att(1, 2)]
        CondL: [att(1, 1) = 'USD']
         SFW-NODE Currencytypes
         ProjL: [att(1, 2), att(1, 1)]
         CondL: []

        JOIN-NODE
        ProjL: [att(2, 1), att(1, 2), att(2, 2), att(2, 3), att(1, 3)]
        CondL: [att(1, 1) = att(2, 4)]
         SFW-NODE Disclosure
         ProjL: [att(1, 1), att(1, 4), att(1, 7)]
         CondL: []

         JOIN-NODE
         ProjL: [att(1, 1), att(2, 1), att(2, 2), att(2, 3)]
         CondL: []
```

```
SELECT
ProjL: [att(1, 2)]
CondL: [att(1, 1) = "DAIMLER-BENZ AG"]
 SFW-NODE Worldscope
 ProjL: [att(1, 1), att(1, 6)]
 CondL: []

JOIN-NODE
ProjL: [att(1, 1), att(2, 1), att(2, 2)]
CondL: []
 SELECT
 ProjL: [att(1, 2)]
 CondL: [att(1, 1) = "DAIMLER-BENZ AG"]
  SFW-NODE Ticker_Lookup2
  ProjL: [att(1, 1), att(1, 2)]
  CondL: []

 JOIN-NODE
 ProjL: [att(2, 1), att(1, 1)]
 CondL: []
  SELECT
  ProjL: [att(1, 2)]
  CondL: [att(1, 1) = "DAIMLER-BENZ AG"]
   SFW-NODE Name_map_Ds_Ws
   ProjL: [att(1, 2), att(1, 1)]
   CondL: []

  SELECT
  ProjL: [att(1, 2)]
  CondL: [att(1, 1) = "DAIMLER-BENZ AG"]
   SFW-NODE Name_map_Dt_Ws
   ProjL: [att(1, 2), att(1, 1)]
   CondL: []
```