Reconciling Equational Heterogeneity within a Data Federation

Aykut Firat, Stuart Madnick, Michael Siegel,
Benjamin Grosof and Frank Manola

Working Paper CISL# 2009-06

April 2009

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E53-320
Massachusetts Institute of Technology
Cambridge, MA 02142

Reconciling Equational Heterogeneity within a Data Federation

Aykut Firat[1], Stuart Madnick[2], Michael Siegel[2], Benjamin Grosof[2] and Frank Manola[3]

[1]*College of Business Administration, Northeastern University, Boston, MA 02115, USA*
[2]*Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[3]*Independent Consultant, Wilmington, MA, USA*

Abstract

Mappings in most federated databases are conceptualized and implemented as black-box transformations between source schemas and a federated schema. This approach does not allow specific mappings to be declared once and reused in other situations. We present an alternative approach, in which data-level mappings are represented independent of source and federated schemas as a network between "contexts". This compendious representation expedites the data federation process via mapping reuse and automated mapping composition from simpler mappings. We illustrate the benefits of mapping reuse and composition by using an example that incorporates equational mappings and the application of symbolic equation solving techniques.

Keywords: Federated DBs, Logic programming, Heterogeneous information, Mediators and Wrappers

1. Introduction

Federated databases offer information integration on demand in dynamic environments, where data warehousing approaches are not feasible [59]. In order to build a federated database system, a federated schema must be created and mappings between the local schemas and this federated schema must be defined. These mappings not only relate fields of the federated and local schemas (schema-level mappings), but also relate the actual data values in them (data-level mappings).

Creating, and maintaining these mappings is one of the most challenging problems of data federation and completely automated solutions do not exist. The goal in this inherently human-assisted process is to reduce the time it takes a human expert to create and maintain these mappings [28]. Schema matching research ([55], [27], [51]), for instance, focuses on building semi-automated tools that help the human expert identify mappings between heterogeneous schemas.

In order to further reduce the time spent in the data federation process, we propose separating data-level mappings from schema-level mappings and defining data-level mappings independent of schemas. This can be accomplished by introducing an extra layer of abstraction in which data-level *conflict-dimensions* are modeled generically, and mappings between *conflict-dimension values* are defined. Each user can then identify his/her relevant context, which we define as specific set of conflict-dimension values; and create his/her view of the federated database. Given the contexts of users and data sources, data translations can be performed using the relevant mappings. Defining data-level mappings independent of schemas allows their reuse, and easier maintenance in the case of data and schema evolution. Further-

more, the actual data-values of the federated schema can be instantly changed based on user demand simply by adopting a different receiver context.

The underlying representation and reasoning for this proposed architecture has been accomplished within a formal logical framework (ECOIN framework), and its feasibility has been validated with a prototype system (ECOIN Mediator) which provides access to both traditional data sources (e.g., relational database systems) and semi-structured information sources (e.g., Web sites). ECOIN framework and mediator is a significant extension of our earlier work ([20, 22]), and introduces the following novelties:

1) Mappings between contexts are represented as a *network* of conversion functions;
2) Complex mappings can be dynamically composed from individually simple component mappings; and
3) Symbolic equation solving techniques are used in reasoning with (i.e. chaining, inverting, and simplifying) a network of equational mappings (i.e. mappings involving mathematical equations)

We have demonstrated the capabilities of ECOIN with several applications that cannot be described here in detail, but will be briefly listed: In an application on *electronic bill presentment and payment* [32], ECOIN was used to integrate information originating from different financial standards such as IFX, OFX and SWIFT; in another ECOIN was applied to *corporate householding* [47] to handle complex and context dependent definitions of corporations with respect to their subsidiaries, branches, divisions, and joint ventures; and in an application on *global comparison aggregation* [66,15], ECOIN was used to resolve conflicts related to currencies, value added tax, shipment inclusions, etc. among several international shopping web sites. Other applications of ECOIN include *counter-terrorism intelligence integration* [48], *supply chain integration* [63], *disaster relief* [41], *travel information integration* [35], *fixed income securities integration* [54], *e-commerce* [16], and *web services integration* [29].

The rest of the paper is organized as follows. Section 2 presents a motivational airfare example in which didactically simple symbolic equation solving is needed for reconciliation of semantic conflicts. The underlying elements of the ECOIN framework are explained, while this simple example is gradually extended throughout the paper. First, representation of this example in the ECOIN framework is described in section 3; then query rewriting and its interaction with symbolic equation solving techniques under the Abductive Constraint Logic Programming framework is explained in Section 4. Before discussing the advantages of our approach in conclusion, we provide an overview of related work and offer a comparison with existing approaches.

2. Motivational Example

Consider the problem of finding cheap airfares on the Web. The actual example in our prototype system uses eight online airfare sites. For didactical reasons, however, we consider the simplified and slightly dramatized scenario shown in Figure 1 having three sources (an airfare source **cheaptickets**, and two ancillary sources **currencyrates** and **cityairport**) and a single receiver (**user**) with conflicting assumptions. We also assume that there is a one-to-one mapping between the federated schema and the local schemas to highlight the data-level conflicts. Surprisingly, even in such a simple scenario the semantic differences provide enough complexity to illustrate some of the important issues.

Under this scenario, we assume that web sites are wrapped as relational databases [13], and the users are presented with a relational database interface. Such a scenario is quite realistic using, for instance, IBM's DB2 Information Integrator. One user of the system, whom we will call Ben, is an international student looking for a round trip ticket from Boston to Istanbul, with departure on June 1st and return on July 1st 2007. Ben wants to obtain the airfare and airline information for his trip and formulates the following SQL query:

Q1: **SELECT** Airline, Airfare
FROM CheapTickets
WHERE DepDate = "01/06/07" and ArrDate= "01/07/07"
and DepCity= "Boston" and ArrCity= "Istanbul";

Ben (and his query), however, has different interpretations of the attribute values. We refer to these different interpretations as a *context*, and here the user and the source have different contexts (shown in Figure 1).

For example, Ben expects to see the bottom-line airfare (round trip, including all taxes and fees), whereas the data source provides separate fares for each direction of travel, *not* including taxes and fees. In addition Ben has to pay a direct air transit visa fee of £27 for each way if the offered flight has a connecting flight from the United Kingdom. Ben works with "dd/mm/yy" style dates, whereas the source assumes US style dates; Ben operates with city names, whereas the source uses airport codes.

As a result of these contextual differences, without any mediation Ben's query would return an empty answer, because cheaptickets uses airport codes instead of city names; and dates are in "mm/dd/yy" format (refer to sample data). Even if these specific differences were dealt with, for example by writing a new query Q2 with changed city codes and date formats (which itself might be a significant challenge for the user, especially if unfamiliar with the details of each of the multiple sources involved):

Q2: **SELECT** Airline, Airfare
FROM CheapTickets
WHERE DepDate = "06/01/07" and ArrDate= "07/01/07" and
DepCity= "BOS" and ArrCity= "IST";

The results returned would be misleading:

Airline	Airfare
British Airways	495
Lufthansa	525

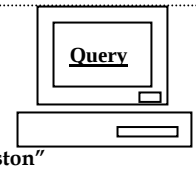
If the original query Q1 were submitted to the ECOIN Mediator, however, the semantic conflicts between the sources and the receiver would be automatically recognized and reconciled (given the necessary context declarations in Figure 1), and Q1 would be rewritten into the following mediated query:

MQ1: SELECT Airline, 2*(Airfare+Tax+
27*exchangeRate) + 25
FROM cheaptickets, currencyrates,
(select Airport from cityairport where city=
"Boston") s1,
(select Airport from cityairport where city=
"Istanbul") s2
WHERE DepDate = "06/01/07" and Arr-
Date="07/01/07" and DepCity = s1.Airport
and ArrCity=s2.Airport and CxnCountry=
"United Kingdom" and fromCur= "GBP" and
toCur= "USD"
and Date= "05/01/07";

UNION
SELECT Airline, 2 * (Airfare+Tax) +25
FROM cheaptickets,
(select Airport from cityAirport where city=
"Boston") s1,
(select Airport from cityAirport where city=
"Istanbul") s2
WHERE DepDate = "06/01/07" and Arr-
Date="07/01/07"
and DepCity= s1.Airport and ArrCity=
s2.Airport and
CxnCountry <> "United Kingdom";

CONTEXT OF BEN

* Fares are expected to be bottom-line price
(round trip, includes taxes, ticket shipment, and transit fees)
* Date is expressed in European style (dd/mm/yy)
* Departure and Destination locations are expressed as city names
* Direct air transit fee of £27 is applied if the plane
has a connecting flight from United Kingdom
* Currency is US dollar
* Today's date: 05/01/07



SELECT **Airline, Airfare**
FROM **cheaptickets**
WHERE **DepDate = "01/06/07" and**
ArrDate= "01/07/07" and DepCity= "Boston"
and ArrCity= "Istanbul";

CONTEXT OF CHEAPTICKETS

* All fares are for each way of travel and do not include fees and taxes.
* Date is expressed in American style (mm/dd/yy)
* Departure and Destination locations are expressed as three letter airport codes
* Ticket shipping cost is $20
* Service fee of $5 is charged
* Currency is US dollars

cheaptickets

ID (I)	Airline (A)	Airfare (F)	Tax (T)	DepDate (DD)	ArrDate (AD)	DepCity (DC)	CxnCountry (CC)	ArrCity (AC)
1	British Airways	495	75	06/01/07	07/01/07	BOS	United Kingdom	IST
2	Lufthansa	525	79	06/01/07	07/01/07	BOS	Germany	IST
...	...	...	...	...	...	...	...	...

CONTEXT OF ANCILLARY SOURCES

Date is expressed in "mm/dd/yy" style

currencyrates

FromCur	ToCur	ExchangeRate	Date
£	$	1.75	05/01/07
…	…	…	…

cityairport

City	Airport
Boston	BOS
Istanbul	IST
…	…

Figure 1 Airfare Example Scenario

In the above mediated query MQ1, in addition to conflicts such as format differences in date and airport codes, conflicts in the interpretation of airfare are also resolved using symbolic equation solving techniques, which will be explained in the later sections. This mediated query is a union of two queries because the airfare calculation depends on whether United Kingdom, which imposes a direct air transit fee for Ben, is part of the flight or not. This information, as shown in Figure 1, is found in Ben's context definition and utilized in query mediation. The first subquery in MQ1 corresponds to the case of having a connecting flight from United Kingdom, and thus adds the transit fee --adjusted in terms of currency-- to the airfare along with tax, shipping and service fees. The airfare is then converted into a round trip airfare (since the airfares reported by cheaptickets are one way fares). As shown by this mediated query, and in the figure, ancillary sources with their own contexts may be automatically added to the query, e.g., in performing data translations such as city to airport code mappings, and currency conversions.

In ECOIN, *query mediation* and *query answering* are clearly separated as in this example. Query mediation is completed with the production of MQ1, and query answering can be accomplished by using the ECOIN POE (planner/optimizer/executioner) [1], which further

processes MQ1 to produce an efficient execution plan, and executes it by submitting subqueries to individual sources. Because MQ1 encodes all the necessary data transformations, the POE system is not concerned with the semantic conflicts between the data sources and users, but simply performs conventional distributed query optimization and execution. MQ1 can also be executed by commercial database systems (such as IBM DB2 Information Integrator) taking advantage of their sophisticated query planners and optimizers. This is one of the main advantages of ECOIN compared to some other query answering systems, which do not separate query rewriting from query answering, thus cannot take advantage of existing commercial database systems. The final results reported by the system below now allow the user to make the right choice and choose Lufthansa over British Airways:

Airline	Airfare
British Airways	1260
Lufthansa	1233

We have only shown one user in this scenario, but other users with different contexts could also pose their queries and get the results in their own context. For example, a different user who wants to compare nominal airfares (one-way, not including taxes and fees) could simply declare or choose her context, issue a query, and get the results in her own context. There is no need to define a new federated schema for this user. Similarly, other data sources can be added to the system with the declaration of their contexts, and queries over multiple sources with different contexts can be mediated in the same way. We will now discuss how this simple scenario can be represented in the ECOIN framework; and how query mediation is accomplished using symbolic equation solving techniques.

3. Representation in the Ecoin Framework

Before the motivational example can work as described in Section 2, knowledge about the user and source contexts needs to be declared. This is accomplished under a formal logical ECOIN framework that has the following five components:

- The *Domain Ontology* **(DO)** is a light-weight ontology – a simple list of object types and their relationships for a domain of interest.
- The *Source Axioms* **(S)** relate sources to the domain ontology and contexts.
- The *Context Ontology* **(CO)** is a description of the object types that can be interpreted differently across data sources and receivers (e.g. Airfare), and the *conflict dimensions* (*modifiers*) for these concepts (e.g. Currency, Inclusion, and Coverage).
- The *Contexts* **(C)** are instantiations of the context ontology. Value assignments are made for each modifier to explicate the meaning of a concept in a data source or receiver (e.g. Currency=USD, Inclusion=nominal, Coverage=one-way).
- The *Conversion Function Network* **(M)** is an organization of primitive conversion functions between different modifier values to achieve conversions between different contexts.

While the underlying formalism of the ECOIN framework is a logical one, system developers are provided with a number of visual tools that support the federated database development process. These graphical meta-data management tools are detailed in [35, 41, 53]. Below we explain some of the components of ECOIN with example logical statements (The full logical declarations for the motivational airfare example can be found in Appendix A).

3.1 Context Ontology (CO)

A context ontology defines how local data models can specialize the generic meanings of the semantic types[1] with the use of a special kind of property called a *modifier*. For each semantic

[1] The types in an ECOIN domain model are called *semantic types*, in that they represent the *generic* semantics of the

type that is interpreted differently by a source or receiver, modifiers are introduced to explicate those differences. For example, in Figure 2 the generic term *airfare* represented by the large cube is specialized along three modification dimensions of *{Coverage, Currency, Inclusion}*. Different values of these modifiers identify the different component cubes of the overall airfare cube, which can be adopted by sources and receivers.

Modifier types themselves can be subject to modification (e.g. how do you represent currency? USD vs. $.) This kind of a difference can be handled via defining modifiers of modifier types. In Figure 2, this situation is illustrated by a *CurrencyFormat* modifier for the *CurrencyType* modifier type, which further slices the smaller lower cube along the currency dimension.

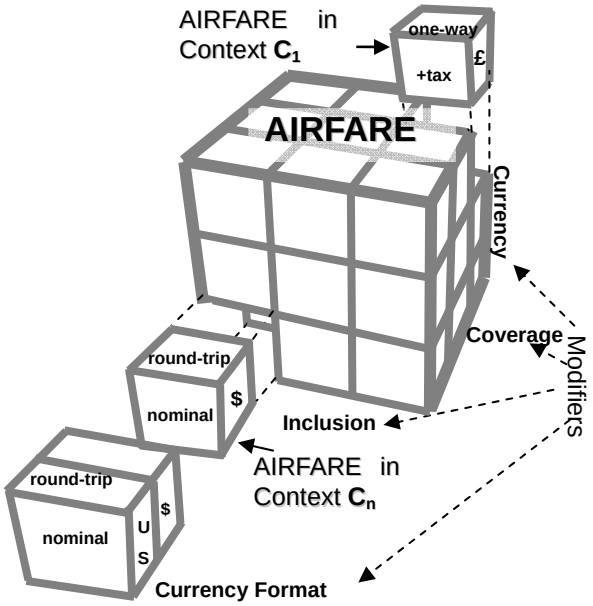


Figure 2. Multi-dimensional Modification of Semantic Types

The fundamental advantage of this approach is that it allows conflicts between sources and receivers to be introduced gradually as they are discovered. Many conflicts emerge later in the federation process as more sources and users are incorporated in the system. When there is a common meaning across the sources and receivers for a given type, no modifier is introduced at that time. When that situation changes at a later time, however, modifiers can be defined to handle the variations.

In Figure 3, we graphically illustrate the complete air-travel ontology its accompanying context ontology. In the Figure terms with multiple meanings across sources and receivers are shown with sliced cubes and corresponding modifiers. Note that airfare and tax types also inherit modifiers of their parent type, moneyAmt.

3.2 Contexts (C)

Context ontology provides a model for specializing generic concepts. Contexts are the actual specializations of the concepts subject to multiple meanings across sources and receivers. For *sources*, contexts define the specializations used for the underlying data values; and for *receivers* contexts describe the specializations assumed in viewing the data values. These specializations may be about the representation of data (e.g. "dd/mm/yy" vs. "mm/dd/yy" style date formats) or nuances in meaning (e.g. nominal vs. bottom-line airfares).

To define a source or receiver context, modifier assignments need to be made. In the underlying logical model the assignments are done via logical predicates. For example, the context labeled as **c_ct** (for **cheaptickets** context) can be described with the following datalog predicates:

currency(MoneyAmt', c_ct, Currency') ← Currency' = object(currencyType, "USD", c_ct, _).
inclusion(Airfare', c_ct, Inclusion') ← Inclusion' = object(inclusionType, "nominal", c_ct, _).
coverage(Airfare', c_ct, Coverage') ← Coverage'=object (coverageType, "one-way", c_ ct, _).
lformat(Airport', c_ct, LFormat')← LFormat' = object(airportFormat, "airport_code", c_ct, _).

---

concepts used in the various data sources.

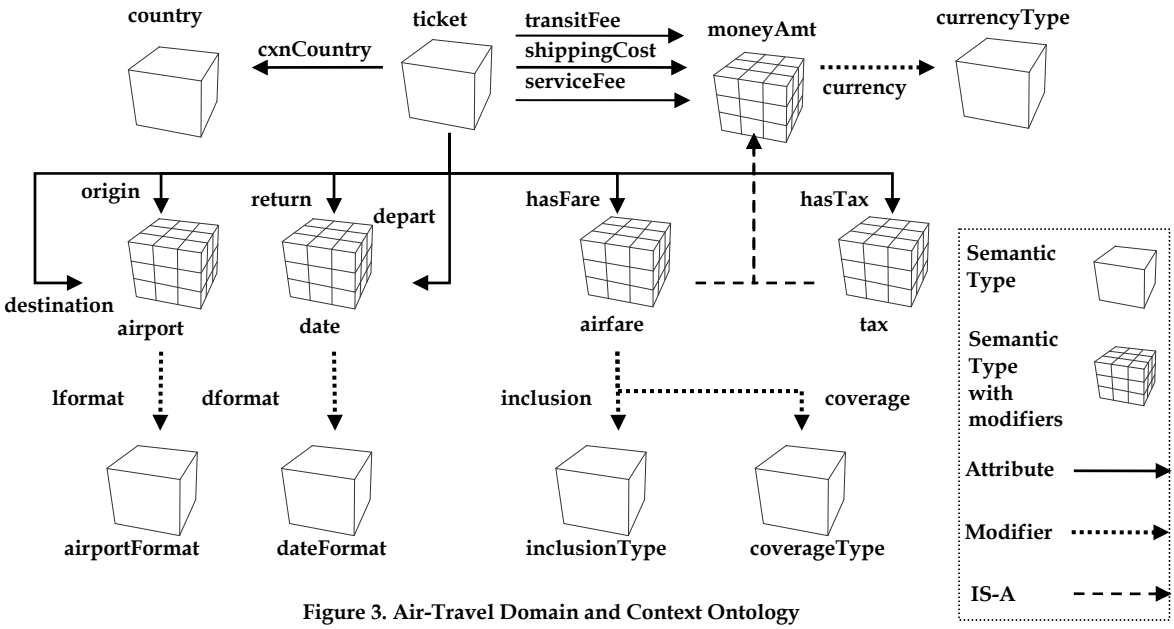


Figure 3. Air-Travel Domain and Context Ontology

dformat(Date', c_ct, DFormat') ← DFormat' = object (dateFormat, "mm/dd/yy", c_ct, _).

These modifier assignments explicitly specify which of the smaller cubes are adopted by the cheaptickets source. The declarations above, for example, simply indicate that *airfare* is *the one-way nominal*[2] *airfare of a ticket in US dollars*, the arrival and departure locations are expressed as airport codes (as opposed to city names), and date is given using the *"mm/dd/yy" style* in the local model of cheaptickets source.

Unlike the above static examples, modifier assignments can also be dynamic. For example, the following declaration for **c_ct** context mean that the currency of transit fees are given in the currency of the connection country and demonstrated the flexibility and power of the underlying logical framework

currency(TransitFee', c_ct, Currency') ← transitFee(Ticket',TransitFee'), cxnCoun try(Ticket',Country'), countryCurrency(Country', Currency').[3]

Contexts may also be organized in a sub-type hierarchy so that values of modifiers are inherited (unless overridden in sub-contexts). With the following modifier assignment for the **MoneyAmt** type in the US context **c_us**, and the subtype relationship between **c_ct** and **c_us**, **Airfare** objects in the **c_ct** context inherit the **currency** modifier value **USD.** This is because **Airfare** is a subtype of **MoneyAmt** and the modifier values of the US context are inherited by the cheaptickets context:

currency(X, c_us, M) ← M = object (currencyType, "USD", c_us, _).
is_a(c_ct, c_us) .

3.3 Conversion Function Network (M)

Mappings in ECOIN ensure that object values in one (source) context can be appropriately mapped to a corresponding value in another (target) context. This is accomplished by defining a conversion function network for each ontological term. Conversion functions are *atomically* defined for each modifier as shown in Figure 4. In this network, conversions between distant nodes can be accomplished by combining individual conversion functions.

[2] Plain value, not including taxes, fees, etc.
[3] Here, countryCurrency is an external relation that relates countries and currencies.

To convert between "nominal" and "+shipping" modifier values, for example, four individual conversions need to be combined.
We now formally define an atomic conversion function for a modifier *m* as follows:

Definition *(Atomic Conversion Function for Modifier m)*

A mapping for modifier m that applies to a semantic type T is a relation P_m: $SO \times PO \times CI \times PO(m) \times PO(m) \times CI \times PO$ where

- *SO is the set of semantic objects of type T from the domain and context ontology (e.g. airfare, date, etc.)*
- *PO is a set of primitive objects (e.g. real, integer, etc.)*
- *PO(m) is a set of primitive objects for modifier m (e.g. one-way, $, etc.)*
- *CI is a domain of context identifiers (e.g. c_ben, c_ct, etc.)*

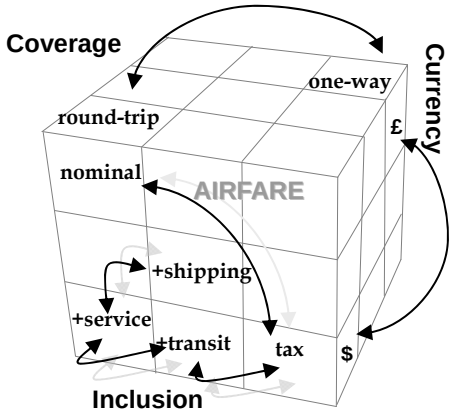


Figure 4. Organization of Conversion Functions for the Ontological term Airfare

For example, the following mapping for the inclusion modifier is defined between the "nominal" and +taxes modifier values:

$P_{inclusion}$(X, VX, SourceContext, "nominal", "+taxes", TargetContext, VY) ← hasFare(T, X), hasTax(T, Tx), value(Tx, SourceContext, VTx), sum(VX, VTx, VY).

In the body, the relationship between source value **VX** and target value **VY** are described as **VX + VTx = VY**, where **VTx** is the corresponding tax value obtained by using the **hasFar**e and **hasTax** predicates. The reserved predicate **value(X,C,Y)** states that the value of semantic object **X** in context **C** is **Y**.
We also define conversion functions as either bidirectional or unidirectional depending on whether their inverse can be used to perform the mapping from target context (TC) to source context (SC). Formally, this condition can be stated as follows:

Definition *(Bidirectional Mappings) A conversion function is bidirectional if*

$P_m(X, \mathbf{VX}, \mathbf{SC}, MVs, MVt, \mathbf{TC}, \mathbf{VY}) \leftrightarrow$

$P_m(X, \mathbf{VY}, \mathbf{TC}, MVt, MVs, \mathbf{SC}, \mathbf{VX})$

The conversion function for the inclusion modifier above, for example, can be declared as bidirectional since it exhibits this symmetry. This can be verified by shuffling the parameters in the header as in the definition, and checking that no change is necessary in the body of the mapping.

Note that the predicates used in the conversion functions, particularly ones like **sum(VX, VTx, VY)**, are *not procedural*; they denote declarative relationships. When translating from **+taxes** to **nominal** using the $P_{inclusion}$ conversion function, for example, **VY** and **VTx** are bound in the equation "VX + **VTx = VY**" and VX cannot be determined procedurally with a summation. It has to be rewritten into "VX = **VY – VTx**" before it can be procedurally calculated. This

is an elementary example, where symbolic manipulation is used for inversion in bidirectional mappings as will be explained in more detail later on.

While the mapping for the inclusion modifier was defined *piecewise*; other mappings can be defined in a non-piecewise manner, as in the example of the currency modifier of the MoneyAmt type:

$P_{currency}$(X, VX, SC, VCurrencyX, VCurrencyY, TC, VY) ← value(Today, SC, VToday), system_date(VToday), value(CurrencyX, SC, VCurrencyX), value(CurrencyY, SC, VCurrencyY), currencyrates'(CurrencyX, CurrencyY, Today, Rate), value(Rate, SC, VRate), mul(VX, VRate, VY).

In this function, the currency values in the source and target contexts, **VCurrencyX** and **VCurrencyY**, are parameterized, and this single function handles all different currency conversions.

Using this network, applicable conversion functions from the network can be obtained by comparing the contexts of the source and the receiver for each requested data type. During this process appropriate compositions should be calculated, and the declarative equations embedded in the conversion functions need to be manipulated to derive the final conversion function. We use the standard shortest path algorithm for composition; and symbolic equation solving techniques to chain, invert and simplify the *equational mappings*. These will be explained in more detail in Section 4.

3.4 The Scalability of the Conversion Function Network

The separation of data-level and schema-level mappings and organizing them as a network has the advantage of scalability as the following analysis shows.

Conversion functions in general can be parametric or non-parametric, and bi-directional or uni-directional. Parametric functions are those that can express the conversion between all possible modifier value pairs with a single rule. Currency conversion function is one such example, because all currency translations can be done using a single rule. Conversion functions for the inclusion modifier, on the other hand, are non-parametric, because a separate conversion function needs to be defined between distinct modifier values in the network. Note, however, we only require the network to be connected; therefore conversion functions need not be defined between all value pairs. If a single function can be used to convert between value pairs (x, y), and (y, x); then that function is called bi-directional; otherwise it is called uni-directional.

Given an ECOIN system with m modifiers, if we assume that

- p percent of the modifiers have parametric functions,
- q percent of the modifiers have bi-directional functions, and
- m modifiers with non-parametric functions have on average θ distinct connections in the network,

the total number of conversion functions used in the system would approximately be:

$$f(m,p,q,\theta) = m(pq+2p(1-q)+(1-p)q\theta+2(1-p)(1-q)\theta)$$
$$= m(2-q)(\theta(1-p)+p)$$

In the airfare scenario, for example, m=5,p=0.6, q=1, and θ=2.5; thus the number of conversion functions are 5(2-1)(2.5(1-0.6)+0.6) = 8, which indeed corresponds to the number of conversion functions in our motivational example fully specified in Appendix A.

In the worst case, when all functions are non-parametric and uni-directional, $f(m,0,0,\theta) = 2m\theta$; and in the best case, when all functions are parametric and bi-directional, $f(m,1,1,\theta) = m$ Thus, in general:

$$\mathbf{m \le f(m,p,q,\theta) \le 2m\theta}$$

Note that the number of conversion functions are not directly proportional to the number of sources and users in the system, although they do have some bearing on the magnitude of m and θ. This relationship is, however, most likely to be logarithmic, as the number of novel conflicts between users and sources would diminish very rapidly with the addition of each new

source and user to the system. In traditional approaches, however, a new conversion routine would need to be established between the mediated schema and each new user and source joining to the system because the architecture does not facilitate the reuse of existing conversion function components. With this brief analysis, we conclude that ECOIN offers the scalability needed for large scale semantic interoperability projects by maximizing the reuse of conversion functions, and minimizing redundancy. (A more detailed comparative analysis can be found in [65], and [67].)

With this analysis, we now move to the query rewriting process involving symbolic equation solving techniques.

4. Query Mediation With Symbolic Equation Solving Techniques

Query answering in ECOIN is a two-step process: first, an intensional answer (mediated query) is returned in response to a user query; then this mediated query can be executed on a conventional query subsystem (e.g. a source's database system) to obtain the extensional answer (actual records). From a practical standpoint, this two-stage process allows us to separate query mediation from query optimization and execution, and thereby take advantage of mature techniques for query optimization in determining how best a query can be evaluated.

In ECOIN, we use symbolic equation solving techniques to derive closed form formulas for data transformations involving equations. Later on, we will discuss how this approach significantly differs from some constraint database approaches, in which database accesses are not interleaved with data transformations. First, however, we describe the task of rewriting an original naïve SQL query (i.e. query with the assumption that no semantic conflicts exist between sources and receivers) as a mediated SQL query (i.e. query with all conflicts between sources and receivers reconciled).

Query mediation in ECOIN is performed via abductive reasoning and constraint processing under the generic framework of Abductive Constraint Logic Programming (ACLP). In the following sub-sections we describe (in order) these steps, which takes a naïve SQL query and rewrites it into a mediated query with source and user conflicts resolved, and the results transformed according to user expectations.

4.1 Naïve To Well-Formed Query Transformation

The queries passed to ECOIN are in SQL, and naïve in the sense that the query is not concerned with the semantic conflicts between sources and receivers. The mediation process starts by trivially converting the SQL query into its naïve Datalog equivalent. For example, the naïve query corresponding to query Q1 in the airfare example of Section 2 is:

answer(Airline,Airfare) ← cheaptickets(I, Airline, Airfare, T, "01/06/07", "01/07/07" , "Boston", C, "Istanbul").

context: c_ben

This naïve query, however, does not correspond to the intentions of the user; and needs to be converted into a well-formed query. In the well-formed query values are contextualized. For example, "Boston" in the original query is expressed with a value predicate instead: **value(City, c_ben, "Boston")**. This means that "Boston" is the value of a semantic object City in the user context **c_ben.** In general, the transformation of a naïve query into a well formed query is accomplished according to the following definition:

Definition *(Naïve to well-formed query transformation)*

Let <Q, c> be a naive query in an ECOIN framework, where c denotes the context from which the query originates and to which the result data must correspond. The well-formed query Q' corresponding to <Q, c> is obtained by replacing all references to the primitive relations with the corresponding semantic relations using distinct variables for each column; and value predicate that maps the variables referred to by the original query to columns in primitive relations.

In the ECOIN framework, primitive relations are sources that are not yet tied to a domain and context ontology. When source columns are annotated with a semantic type and a context, they are called semantic relations (See [12, 21] for more details). Using this transformation on our example, the naïve datalog query is replaced by the following well-formed datalog query:

answer(Airline,Airfare) ← cheaptickets' (X1, X2, X3, X4, X5, X6, X7, X8, X9),
value(X2, c_ben, Airline), value(X3, c_ben, Airfare),
value(X5, c_ben, "01/06/07"), value(X6, c_ben, "01/07/07"),
value(X7, c_ben, "Boston"), value(X9, c_ben, "Istanbul").

4.2 Transformation Between Ecoin And ACLP

Our predecessor COIN used *abductive logic programming* (ALP), which is an extension of logic programming (LP) [45] as a reasoning technique for query mediation. Abductive reasoning, in its simplest case takes the form:

From observing A and the axiom B → A
Infer B as a possible "explanation" of A.

Here the set of facts that can be used as explanations are called abducibles (e.g. B). We build on top of the COIN reasoning technique, by using an extension of ALP with constraint logic programming, (i.e. ACLP) in ECOIN. Integration of ALP and constraint logic programming (CLP) frameworks has been pursued by ([34], [33]) based on the view that they can both be understood within the same conceptual framework of hypothetical reasoning. In both frameworks, an answer to a query is constructed from special predicates (i.e. abducible predicates in ALP; constraint predicates in CLP) which are constrained either by integrity constraints in the case of ALP or by means of a constraint theory in CLP. In ECOIN, we use ACLP, to unify the treatment of abducibles (relations) and constraints (data transformation formulas) during query mediation.

Formally, ACLP can be seen as an extension of the ALP framework that supports constructive abduction allowing the abducible hypotheses to take the non-ground form of ∃X (A(X), C(X)) where A is a conjunction of abducible atoms and C is a set of constraints defined over the CLP (arithmetic) domain. An ACLP framework can be defined as follows:

Definition *(ACLP Framework)*

An abductive CLP (or ACLP) theory is a triple (P, A, IC) where

- *P is a constraint logic program.*
- *A is a set of abducible predicates different from the constraint predicates.*
- *IC: is a set of closed first order formulae (Integrity Constraints) over the combined language of CLP and P.*

In order to transform ECOIN framework into an ACLP framework we define the following mapping:

Definition *(ECOIN to ACLP transformation)*

An ECOIN framework (DO, CO, C, S, M) can be mapped to a corresponding ACLP framework given by (P, IC, A) where

- *P is the Datalog^{neg} [4] translation of the set of clauses in (DO, CO, C, S, M)*
- *IC consists of source integrity constraints, augmented with* ***symbolic equation solving constraint theory***

[4] Datalog with negation

- *A consists of non-ground and ground extensional predicates defined in S, the built-in predicates corresponding to arithmetic and relational (comparison) operators, and the system predicate which provides the interface for system calls.*

Note that the transformation between ECOIN and ACLP requires the augmentation of the integrity constraints with ***symbolic equation solving constraint theory***. This is explained in more detail next.

4.3 Symbolic Equation Solving Constraints

The purpose of symbolic equation solving in ECOIN is to convert declarative predicates such as **sum(X, Y, Z)**, or **mul(X, Y, Z)** into procedural query expressions. In a bidirectional mapping, for example, **sum(VX, VTx, VY)** with bound variables **VTx** and **VY** needs to be converted into a procedural expression to compute the result **VX**, (i.e., to **Y – Tx).** Furthermore, translations between contexts may be composed from multiple conversion functions, which may necessitate solving simultaneous arithmetic expressions. Consider for example, **sum(VX, VTx, VY)** and **mul(**VZ**, VRate, VX)** together, in which the bold variables are bound to some unknown value. In order to express **VZ** in the procedural form **(VY – VTx) / VRate,** we need to perform arithmetic inversions and combinations**.** Finally, if the expressions involve numeric values instead of variables, they may be simplified by evaluating those expressions during query mediation. For example, **sum(X1, X2, X3)** is a basic arithmetic constraint predicate corresponding to: **X3 = X1 + X2**, where **X3** is the result variable. Thus, given sum(X, 5, 20), we may replace X with 15 before submitting the mediated sub-queries to local data sources. Using ACLP with symbolic equation solving constraints, all of these can be performed automatically through logical resolution and constraints.

In our current framework we only allow *basic arithmetic constraint predicates*, which have been found to be sufficient for many cases and are defined as follows:

Definition *(Basic Arithmetic Constraint Predicate)*

*A basic arithmetic constraint predicate is a predicate of arity 3 corresponding to arithmetic operators {+, -, *, /, ** [5]}. The third variable is called the result variable.*

By using basic arithmetic predicates, polynomial expressions of arbitrary complexity can be constructed. For example:

$$f(x, y, z) = x^2y + z/2 - 10$$

can be constructed by the following combination of basic arithmetic predicates:

pow(X,2,R1), mul(R1,Y, R2), div(Z,2,R3), sum(R2, R3, R4), sub(R4,10, F)

Symbolic equation solving constraints are encoded declaratively in the Constraint Logic Programming (CLP) paradigm by using Constraint Handling Rules (CHR) [17]. We first provide some background on CLP and CHR, before describing the symbolic equation solving constraints.

Constraint logic programming and Constraint handling rules (CHR)

Constraint logic programming (CLP) extends ordinary logic programming (LP) with constraint predicates, which are checked for satisfiability and simplified by means of a constraint solver. Like LP, a CLP program needs to search a database of facts, but it can use constraints to rule out many possible outcomes and prune away large parts of the search tree.

Although constraint solvers were originally black box systems, languages such as *constraint handling rules* (CHR) allow users to write their own constraint solvers in a high level language. CHR has been used to construct a wide range of solvers including terminological and temporal reasoning. Below we summarize the syntax and semantics of CHR, borrowing material

[5] ** stands for exponent

from [17]. The purpose here is to set the background for the reader to understand the declarative encoding of symbolic equation solving constraints, which is described right afterwards.

Syntax of CHR

Definition: (CHR Program)

A CHR program is a finite set of CHR. There are two basic kinds of CHR.

A simplification CHR has the form:

H1,..., Hi <=> G1,…, Gj | B1, …, Bk,

A propagation CHR has the form

H1,..., Hi ==> G1,…, Gj | B1, …, Bk,

with $i > 0$, $j \geq 0$; $k \geq 0$ and where the multi-head H1,..., Hi is a nonempty sequence of CHR constraints, the guard G1,…, Gj is a sequence of built-in constraints, and the body B1, …, Bk is a sequence of built-in and CHR constraints. Declaratively, CHR relates heads and body provided the guard is true.

Below is an example set of CHR for simplification, propagation, and the use of guards:

reflexivity @ **X =< Y <=> X=Y | true.**
antisymmetry @ **X=<Y, Y=<X <=> X=Y.**
transitivity @ **X=<Y, Y=<Z ==> X=<Z.**

The first rule replaces X=<Y with true (an empty sequence) provided that X=Y. Thus whenever the constraint solver encounters the constraint X=<X it is removed from the constraint store. The second rule means that whenever we find both X=<Y and Y=<X in the current constraint we can replace them with the logically equivalent X=Y. Finally, transitivity adds the new redundant constraint, X=<Z, to the store whenever it encounters both X=<Y and Y=<Z in the current constraint. This new constraint, although redundant, may activate other rules in the constraint store and achieve useful simplifications.

Symbolic Equation Solving Constraint Theory

In ECOIN, query mediation and query answering are separated, thus constraints are not used to process actual records from the databases, but to manipulate arithmetic predicates such that all variables are bound; therefore all expressions are computable by an external query optimizer and execution engine at run time. To explain the details, we first define the notion of boundness:

Definition *(Boundness)*

- *A variable is bound if it is implicitly or explicitly bound.*
- *A variable that references a data element in a source relation is implicitly bound.*
- *A variable that references a ground atom (e.g. a numerical value) is explicitly bound.*
- *A variable X that is functionally dependent on a set of variables S (shown as $S \rightarrow X$) is implicitly bound if all elements of S are bound.*

For example,

- in cheaptickets (X1, X2, X3, X4, X5, X6, X7, X8, X9) each Xi (i=1..9) is implicitly bound.
- In {X3 = 500, X4=50}, X3, and X4 are explicitly bound.

- In {X3 = 500, X4=50, X10 = X3 + X4}, X10 is implicitly bound since {X3, X4} $\rightarrow$ X10 and X3, X4 are bound.

Now, the definition of Symbolic equation solving constraint theory can be given:

Definition *(Symbolic Equation Solving Constraint Theory for Arithmetic Predicates)*

*Symbolic equation solving constraint theory for arithmetic predicates, SCT(A), is a CHR program defined for a set of arithmetic predicates A corresponding to arithmetic operators {+, -, *, /, **}. SCT(A) reduces a given goal store G, ($p \in G \rightarrow p \in A$), to a constraint store G′ ($p \in G' \rightarrow p \in A$) such that $\forall$ pi(vi1, vi2, vi3) $\in$ G′, vi1, vi2, vi3 are bound if such a reduction exists.*

For example,

- Suppose {X4, X2} are implicitly bound, then SCT({sum, mul, div, sub, pow}) would reduce G = {sum(X1, X2, X3), mul(X3, .15, X4) } to G′ ={sub(X3, X2, X1), div(X4, .15, X3) } where {X4, .15} $\rightarrow$ X3 $\wedge$ {X3, X2} $\rightarrow$ X1 establishes that all variables are now bound.

In Appendix B, we provide a few constraint examples G of SCT(A) to illustrate the declarative and extendable nature of the framework. They are generic, thus part of the system, and not defined differently for each application.

4.4 The Abduction Step

In the next mediation step, the well formed query is processed in the ACLP framework with standard abduction augmented with external calls to custom sub-programs (e.g., the use of Dijkstra's shortest path algorithm in determining the shortest paths in the mapping network). After the well-formed query goes through the abductive reasoning, the following constraint predicates are posted to the constraint store:

{{ answer(VAirline,VAirfare), cheaptickets(I,VAirline, Airfare, T, "06/01/07", "07/01/07", Airport1, "United Kingdom", Airport2),
cityairport("Boston", Airport1), cityAirport("Istanbul", Airport2),
currencyrates("GBP","USD", ExchangeRate, "05/01/07"), sum(Airfare, T, PT),
mul(ExchangeRate,27,VFA), sum(PT,VFA,PTV), sub(PTVS,5,PTV), sub(Final, 20, PTVS),
sum(20,5,SP),sub(Final, SP, FSP), div(RT,2,FSP), sum(RT,SP,VAirfare).
}
{ answer(VAirline,VAirfare), cheaptickets(I,VAirline, Airfare, T, "06/01/07", "07/01/07", Airport1, Cxn, Airport2), Cxn <> "United Kingdom",cityAirport("Boston", Airport1), cityAirport("Istanbul", Airport2), sum(Airfare, T, PT), sum(PT,0,PTV), sub(PTVS,5,PTV), sub(Final, 20, PTVS),sum(20,5,SP), sub(Final, SP, FWOP), div(RT,2,FWOP), sum(RT,SP,VAirfare).}}

which are further processed as illustrated in Appendix C to produce the mediated query **MQ1** of section 2.

Thus by only using declarative knowledge about the integration domain formalized by the ECOIN framework, and a simple set of symbolic equation solving constraint rules, we are able to mediate queries using a standard inference procedure ACLP.

5. Comparison With Existing Approaches

Over the last two decades there have been numerous studies on database and information integration under a variety of titles such as multidatabase systems ([43], [40]), heterogeneous database systems [36,37], mediators [64], answering queries using views [26], and federated information systems [8]. While there have been advances in *physical* integration of disparate data sources, (e.g. IBM's DB2 Information Integrator is able to provide a SQL query interface to data from disparate data sources including XML documents, Web Services, programs such as Blast, and so on), there is still much to do to achieve *semantic* integration (i.e. combining data meaningfully).

ECOIN, like its predecessor, is a combination of ideas derived from different threads in the literature in artificial intelligence (on “contexts" [24, 25]), databases (on “heterogeneous databases" [56, 57, 60, 21]), logic programming [39, 7, 38], and on “abductive logic programming" [33]. ECOIN has been designed with the observation that semantic information integration should have the dual purpose of: 1) *reconciling* semantic heterogeneity across information *sources*; and 2) *supporting* semantic heterogeneity across information *receivers* to allow the receivers to use their own, heterogeneous semantics in spite of what the sources support. In traditional tightly coupled approaches (e.g. Information Manifold [42], DISCO [62], SIMS [3], etc.) the first purpose is achieved via mappings between local sources and a federated schema; yet the users are locked into a single system-defined integrated view (federated view), which violates the second purpose. In loosely coupled approaches (e.g. VIP-MDBMS [40], MRDSM [44], TSIMMIS [19]), the second goal is naturally achieved as every user formulates queries to get results in their own view; but nothing is done at the system level to achieve the first goal. With ECOIN, we satisfy both purposes by associating every information source and receiver with a *context*.

The literature on semantic interoperability is very large, and is getting even larger with the emergence of the Semantic Web ([6]) as a giant semantic interoperability project ([50], [5], [30]); therefore we will not be able to provide a complete review of the related projects, but instead summarize the major advantages or differences of ECOIN over existing approaches under a couple of headings as follows:

The Scalability of the Mappings

There is substantial work on automatically finding mappings between schemas using machine learning techniques such as corpus-based schema mapping [46], automatic schema matching algorithms ([58], [18]), and schema mapping as discovery approaches [52]. We consider all of these complementary to ECOIN, because we can utilize the discovered mappings in our conversion function network. In most of these approaches, however, data-level mappings are defined between source schemas and a mediated schema; and in some emerging architectures that favor local peer-to-peer data translations mappings are defined directly between source schemas as in Piazza [31], and Hyperion [4]. In ECOIN, we go one step further by defining data level mappings independent of schemas.

This abstraction presents a number of opportunities for systematic sharing and reuse of semantic mappings. For example, different sources and receivers in the same context may bind to the same set of context axioms; and appropriate mappings would automatically apply. Because of this potential to share conversion functions, defining mappings as a network between contexts becomes more economical than defining them between sources and the target schema when there are large numbers of disparate sources. This has been demonstrated with the analysis in Section 3.4.

In addition, if source semantics change in the future, nothing needs to be done in most cases other than updating context declarations. In the worst case, when unforeseen semantic conflicts emerge with the addition of new sources, the domain and context ontology needs to be updated with the introduction of new types and modifiers, sources and receivers need to define their values for the new modifiers, and conversion function network should adopt new functions to extend the network to cover the new modifiers. While this requires some new work, the complexity of creating and administering (maintaining) semantic interoperation services does not increase exponentially with the number of participating sources and receivers, and changes can be incorporated in a graceful manner. The same situation would require substantial amount of work in the tightly coupled systems as the mappings between the sources and the mediated schema would need to be completely redefined.

Separation of Query Rewriting and Answering

Another area of research investigates the use of constraints in data integration [11], and query optimization [10]. The major difference of our work with this line of research is that we separate **query rewriting** from **query answering**, and produce an intermediate mediated query. The constraints used in our approach encode symbolic equation solving rules, which are used to manipulate query predicates, not actual data. The constraint-based approach explained in [11], on the other hand, mixes database accesses with data transformations thus does not consider query rewriting at all. Constraints in these approaches are stored in a central constraint database, and apply to actual data that needs to pass through the constraint database before being returned to the user; no intermediate query is produced.

By keeping query rewriting and answering separate not only do we provide the user with an intermediate intensional answer that may be used to confirm the user's understanding of what the query actually entails, but also take advantage of mature techniques for query optimization in determining how best a query can be evaluated. ECOIN, for example, can easily be plugged in to a commercial distributed database system. It would intercept a user query, rewrite it into a mediated query, and pass it back to the database system, which would then use its state of the art distributed query optimization and execution techniques to obtain answers. The same thing cannot be accomplished by systems which only address query answering. Those systems lack transparency as they are unable to account for the needed transformations, and would need to develop systems that incorporate mature optimization techniques before becoming practical solutions.

Understanding this distinction is crucial before comparing other constraint based approaches with ours. Along the same lines, equation solving used in MRDSM [46] falls under the same category, as its data transformations were not concerned with query rewriting but query answering alone.

Multi-Faceted Federated Schema

The majority of traditional database integration approaches based on schema-matching techniques ([58]) provide a uniform query interface in the form of a federated schema because the users may not be familiar with the sources to formulate their queries, or may not know the contents and coverage of the data sources. In ECOIN, a federated schema is treated like any other source with its own context. In ECOIN, users can query the same federated schema yet get the data in their own context. For example, two users could use the same column airfare of the federated schema in their queries, but one could see the airfare with taxes in dollars, and the other would see the airfare without taxes in Euros in accord with their contexts. Supporting multiple integrated views over a single federated schema is a major advantage of ECOIN compared to traditional approaches.

ECOIN and the Semantic Web

The Semantic Web is the vision of achieving semantic interoperability on the Internet ([6]). It differs from older application environments in many ways, but particularly in its huge number of autonomous sources and the rapid and continuous change these sources are going through ([49]). So far, the Semantic Web seems to approach the problem of semantic interoperability in much the same way that earlier work on semantic data integration did, using ontologies (schemas) and rules, without the use of modular contexts and conversion functions a la ECOIN. In order to translate the conceptual framework of ECOIN to that of the Semantic Web, further work is needed to find out how to represent contexts, conversion function networks using the available Semantic Web constructs such as ontologies, and rules [23]. Preliminary results of such an effort can be found in [61].

6. CONCLUSION

We have presented in this paper ECOIN, which offers an alternative approach to reconciling semantic heterogeneity in federated database systems. ECOIN separates data-level mappings from schema-level mappings and represents and reasons with mappings as a network between contexts. By reducing the number of conversion functions needed to a minimum, we are able to reduce time spent in building a federated database application.

We have also demonstrated a novel integration of symbolic equation techniques in reasoning with equational mappings by intertwining query mediation with abductive constraint logic programming. This study, to our knowledge, is the first example of *query rewriting* by symbolically manipulating equational mappings. We have implemented these ideas in a prototype implementation [12] using the Eclipse Prolog engine [9] and procedural programming languages. This prototype provides mediated access to traditional databases, as well as semi-structured web sites, and web services; creates and maintains metadata that are used in ECOIN through graphical interfaces, and supports merging multiple applications [14]. With several real life application demonstrations and sound theoretical basis, ECOIN provides an elegant solution for dynamic reconciliation of semantic conflicts.

ACKNOWLEDGMENT

This research was supported, in part, by The MITRE Corporation, Merrill Lynch, PriceWaterHouseCoopers, Suruga Bank, MIT-Singapore and MIT-Malaysia Alliances.

REFERENCES

[1] ALATOVIC, T. (2001) Capabilities Aware, Planner, Optimizer, Executioner for Context Interchange Project. Master of Engineering Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

[2] ANWAR, F. 2004 "Virtual Merging of Ecoin Applications and Guidelines for Building Ontologies", Masters of Engineering Thesis, Massachusetts Institute of Technology, June, 2004.

[3] ARENS, Y., KNOBLOCK, C., SHEN, W. (1996). Query Reformulation for Dynamic Information Integration. Journal of Intelligent Information Systems 6 (2/3): 99-130.

[4] ARENAS, M., KANTERE, V., KEMENTSIETSIDIS, A., KIRINGA, I., MILLER, R., AND MYLOPOULOS, J. 2003. The Hyperion Project: From Data Integration to Data Coordination. In SIGMOD Record, Special Issue on Peer-to-Peer Data Management, 32(3):53-58.

[5] BAADER, F., HORROCKS, I., SATTLER, U. 2003. Description Logics as Ontology Languages for the Semantic Web. Festschrift in honor of J. Siekmann, Springer, 2003.

[6] BERNERS-LEE, T., HENDLER, J., LASSILA, O. 2001. The Semantic Web. Scientific American, May 2001 Issue.

[7] BURCHERT, H. J. 1994. A Resolution Principle for Constrained Logics, Artificial Intelligence, 66, 235--271, 1994.

[8] BUSSE, S., KUTSCH R. D., AND LESER, U. 1999. Federated Information Systems: Concepts, Terminology and Architectures, Technical Report Nr. 99-9, TU Berlin.

[9] CHEADLE, A. M., HARVEY, W., SADLER, A.J., SCHIMPF, J., SHEN, K., AND WALLACE M. G., 2003. ECLiPSe: An Introduction by. IC-Parc, Imperial College London, Technical Report IC-Parc-03-1.

[10] CHEN, L., 2003. Using constraints to describe source contents in data integration systems, IEEE Intelligent Systems, vol:18:5, pp: 49- 53, Sept/Oct 2003.

[11] CHENG, X., DONG., G., LAU, T., SU, J, 1999. Data Integration by Describing Sources with Constraint Databases, In Proc. Int. Conf. on Data Engineering (ICDE).

[12] FIRAT, A. 2003. Information Integration using Contextual Knowledge and Ontology Merging, MIT Ph.D. Thesis.

[13] FIRAT, A., MADNICK, S., AND SIEGEL, M. 2000. The Caméléon Web Wrapper Engine, In Proceedings of the VLDB2000 Workshop on Technologies for E-Services, 1-9.

[14] FIRAT, A., MADNICK, S., AND GROSOF, B. 2006. Contextual Alignment Of Ontologies In The Ecoin Semantic Interoperability Framework, forthcoming.

[15] FIRAT, A., KALEEM, M., LEE, P., MADNICK, S., MOULTON, A., SIEGEL, M., ZHU, H. (2003) "COntext INterchange (COIN) System Demonstration", Proceedings of the Semantic Integration Workshop, Sanibel Island, Florida, October 20, 2003.

[16] FIRAT, A., MADNICK, S., AND GROSOF, B. 2002 Financial Information Integration in the Presence of Equational Ontological Conflicts, 12th Workshop on Information Technology and Systems, Barcelona, Spain, 2002.

[17] FRÜHWIRTH, T. 1998. Theory and Practice of Constraint Handling Rules, Special Issue on Constraint Logic Programming (P. Stuckey and K. Marriot, Eds.), Journal of Logic Programming, Vol 37(1-3), pp 95-138, October.

[18] GAL, A., ANABY-TAVOR, A., TROMBETTA, D., AND MONTESI, D. 2005, A Framework for Modeling and Evaluating Automatic Semantic Reconciliation, VLDB Journal, vol. 14:1, pp. 50-67.
[19] GARCIA-MOLINA, H., HAMMER, J., IRELAND, K., PAPAKONSTANTINOU, V., ULLMAN, J., WIDOM, J. (1995). Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford,California, March 1995.
[20] GOH, C. H., MADNICK, S. E., AND SIEGEL, M. 1994. 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, pp. 337-346, Gaithersburg, MD.
[21] GOH, C. H. 1997. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, MIT Ph.D. Thesis.
[22] GOH, C.H., BRESSAN, S., MADNICK, S., AND SIEGEL, M. 1999. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. ACM Transactions on Information Systems, 17: 3, 270–293.
[23] GROSOF, B. 2001 "Representing E-Business Rules for the Semantic Web: Situated Courteous Logic Programs in RuleML", In Proceedings of the Workshop on Information Technologies and Systems ,WITS '01.
[24] GUHA, R.V. 1990. Micro-theories and contexts in cyc part i: Basic issues. Technical Report MCC Technical Report No. ACT-CYC-129-90, MCC Microelectronics and Computer Technology Corporation.
[25] GUHA R. V. (1991). Contexts: a formalization and some applications, MCC Tech Rep ACT-CYC42391.
[26] HALEVY, A. (2000). Theory of Answering Queries Using Views. ACM SIGMOD Record 29(4): 40-47.
[27] HALEVY, A.Y., MADHAVAN, J., AND BERNSTEIN, P. A. 2003. Discovering Structure in a Corpus of Schemas, Data Engineering Bulletin, September 2003, 26-33.
[28] HALEVY, A. (2005). Why Your Data Won't Mix. ACM Queue, 3:88, 50-58, 10/2005.
[29] HANSEN, M., MADNICK, S., SIEGEL, M. 2002. Data Integration using Web Services. DIWeb 2002: 3-16.
[30] HORROCKS, I., TESSARIS, S. 2002: Querying the Semantic Web: a Formal Approach. In: Proc. of the 13th Int. Semantic Web Conf. ISWC '02.
[31] IVES, Z., HALEVY, A., MORK, P., AND TATARINOV, I. 2004. Piazza: Mediation and Integration Infrastructure for Semantic Web Data. Journal of Web Semantics, Vol. 1 No. 2, February 2004, 155-175.
[32] JAYASENA, S., BRESSAN, S., AND MADNICK, S. 2005. A case study of Electronic Bill Presentment and Payment (EBPP) integration using the COIN mediation technology, Journal of Internet Banking and Commerce, Summer 2005, vol. 10, no. 2
[33] KAKAS, A. C., MICHAEL, A., AND MOURLAS, C. 2000. ACLP: Abductive Constraint Logic Programming, Journal of Logic Programming, 44(1-3):129-177.
[34] KAKAS, A. C., AND MICHAEL, A. 1995. Integrating abductive and constraint logic programming. In Proc. International Logic Programming Conference.
[35] KALEEM, M. 2003. CLAMP: Application Merging in the ECOIN Context Mediation System using the Context Linking Approach, MIT Master's Thesis, EECS.
[36] KASHYAP, V. AND SHETH, A. P. 1996. Semantic and schematic similarities between database objects: A context-based approach. VLDB Journal: Very Large Data Bases, 5(4):276--304.
[37] KASHYAP, V. AND SHETH, A. P. 2000 Information Brokering across Heterogeneous Digital Data: A Metadata-based Approach, Kluwer Academic Publishers.
[38] KIFER, M., LAUSEN, G., AND WU, J. 1995. Logical foundations of object-oriented and frame-based languages. Journal of ACM 42, 4 (July 1995), 741–843.
[39] KOWALSKI, R. A. 1992. A dual form of logic programming. Lecture Notes, Workshop in Honour of Jack Minker, University of Maryland, November.
[40] KUHN, E., LUDWIG, T. (1988). VIP-MDBS: a logic multidatabase system, In Proceedings of the first international symposium on Databases in parallel and distributed systems,p.190-201, December 05-07, Austin, Texas, United States.
[41] LEE, P. 2003. Metadata Representation and Management for Context Mediation, MIT Master's Thesis, EECS.
[42] LEVY, A. 1998. "The Information Manifold Approach to Data Integration," IEEE Intelligent Systems, 1312-16.
[43] LITWIN, W., AND ABDELLATIF, A. 1986. Multidatabase interoperability. IEEE Computer, vol. 19:12, pp. 10-18.
[44] LITWIN, W., AND ABDELLATIF, A. 1987. An overview of the multi-database manipulation language MDSL. In Proceedings of the IEEE, 75(5):621-632.
[45] LLOYD, J. W. 1987. Foundations of logic programming. Springer-Verlag, 2nd, extended edition.
[46] MADHAVAN, J., BERNSTEIN, P. A., CHEN, K., HALEVY, A., AND SHENOY, P. 2003. Corpus-based Schema Matching. Workshop on Information Integration on the Web, at IJCAI'2003, pp. 59-66.
[47] MADNICK, S, WANG, R., AND XIAN, X. 2004. The Design and Implementation of a Corporate Householding Knowledge Processor to Improve Data Quality, Journal of Management Information Systems, Vol. 20 No. 3, Winter 2004 pp. 41 - 69
[48] MADNICK, S., MOULTON, A., AND SIEGEL, M. 2005. Context Mediation Demonstration of Counter-Terrorism Intelligence (CTI) Integration, MIT Sloan School of Management, Working Paper.
[49] MANOLA, F. 2002. The Semantic Web and the Role of Information Systems Research, NSF-OntoWeb Invitational Workshop on DB-IS Research for Semantic Web and Enterprises.

[50] MCGUINNESS, D. L., FIKES, R., HENDLER, J., AND STEIN, L. A. 2002. DAML+OIL: An Ontology Language for the Semantic Web. IEEE Intelligent Systems, Vol. 17, No. 5, pages 72-80, September/October 2002
[51] MELNIK, S. 2004. Generic Model Management: Concepts and Algorithms, Springer-Verlag.
[52] MILLER, R., HAAS, L., HERNANDEZ, M. 2000. Schema Mapping as Query Discovery, In Proceedings of the 26th VLDB Conference, Cairo, Egypt, 77-88.
[53] Usman Y. Mobin Graphical Metadata Management for the Context Mediation System, Master of Engineering Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.
[54] MOULTON, A., MADNICK, S., AND SIEGEL, M. 2002. Semantic Interoperability in the Securities Industry: Context Interchange Mediation of Semantic Differences in Enumerated Data Types. DEXA Workshops 2002: 883-888.
[55] RAHM, E., BERNSTEIN, P. 2001. A Survey of Approaches to Automatic Schema Matching, The VLDB Journal, 10:334-350.
[56] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. (1992). Context interchange using metaattributes, In Proceedings of the 1st International Conference on Information and Knowledge Management, pages 377-386.
[57] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. 1994. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems, ACM Transactions on Database Systems, 19(2):254-290.
[58] SHETH, A., KASHYAP, V. 1992. So Far (Schematically) Yet So Near (Semantically), In Proceedings of the IFIP TC2/WG 2.6 Conference on Semantics of Interoperable Database Systems, DS--5, Lorne, Victoria, Australia.
[59] SHETH, A. P. AND LARSON, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3):183-236.
[60] SIEGEL, M., AND MADNICK, S. 1991. A Metadata Approach to Resolving Semantic Conflicts, In Proceedings of the Seventeenth International Conference on Very Large Databases, Barcelona, Spain, Sept 1991, VLDB Endowment, Berkeley, CA, 133–145.
[61] TAN, P., MADNICK, S., TAN, K. 2004 Context Mediation in the Semantic Web: Handling OWL Ontology and Data Disparity through Context Interchange, MIT Sloan Working Paper No. 4496-04; CISL Working Paper No. 2004-13.
[62] TOMASIC, A., RASCHID, L., AND VALDURIEZ, P. (1998). Scaling access to heterogeneous data sources with disco, IEEE Transactions on Knowledge and Data Engineering, 10(5):808-823.
[63] TU, S., MADNICK, S., WU, L. 2004.Improving Uccnet-compliant B2B Supply-chain Applications on a Context Interchange Framework, MIT Sloan School of Management Working Paper;4478-04
[64] WIEDERHOLD, G. "Mediators in the Architecture of Future Information Systems"; IEEE Computer, March 1992, pages 38-49
[65] ZHU, H., MADNICK, S., 2004. Context Interchange as a Scalable Solution to Interoperating Amongst Heterogeneous Dynamic Services, MIT Sloan School of Management, Working Paper CISL# 2004-16.
[66] ZHU, H. MADNICK, S., SIEGEL, M. 2002 "Global Comparison Aggregation Services", Proceedings of the 1st Workshop on E-Business, Barcelona, Spain.
[67] ZHU, H. 2005. "Effective Information Integration and Reutilization: Solutions to Technological Deficiency and Legal Uncertainty", MIT Ph.D. Thesis.

APPENDIX A

ECOIN DECLARATIONS FOR THE AIRFARE EXAMPLE (FULL)

DOMAIN ONTOLOGY

Types:

semanticType(basic), semanticType(country), semanticType(ticket), semanticType(moneyAmt), semanticType(currencyType),
semanticType(airport), semanticType(date), semanticType(airfare), semanticType(tax), semanticType(airportFormat),
semanticType(dateFormat), semantiType(inclusionType),
semanticType(coverageType).

Type hierarchy:

isa(airfare, moneyAmt). isa(tax, moneyAmt).

Attributes:

cxnCountry(ticket, country), transitFee(ticket, moneyAmt), shippingCost(ticket, moneyAmt), serviceFee(ticket, moneyAmt), origin(ticket, airport), destination(ticket, airport), return(ticket, date), depart(ticket, date), hasFare(ticket, airfare), hasTax(ticket, tax).

SOURCES

Source Elevations:

cheaptickets'(ID, Airline, Airfare, Tax, DepDate, ArrDate, DepCity, CxnCountry, ArrCity)
ID = object(ticket, I, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
Airline = object(basic, A, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
Airfare = object(airfare, F, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
Tax = object(tax, T, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
DepDate = object(date, DD, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
ArrDate = object(date, AD, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
DepCity = object(airport, DC, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
CxnCountry = object(country, CC, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)),
ArrCity = object(airport, AC, c_ct, cheaptickets(I,A,F,T,DD,AD,DC,CC,AC)).

currencyrates'(FromCur, ToCur, Rate, Date) ←
FromCur= object(basic, F, c_ancillary, currencyrates(F,T,R,D)),
ToCur= object(basic, T, c_ancillary, currencyrates(F,T,R,D)),
Rate= object(basic, R, c_ancillary, currencyrates(F,T,R,D)),
Date=object(date, D, c_ancillary, currencyrates(F,T,R,D)).
cityairport'(City, Airport) ←
City = object(basic, C, c_ancillary, cityairport(C,A)),
Airport = object(basic, A, c_ancillary, cityairport(C,A)).

Attribute Assignments:

cxnCountry(T, C) ← cheaptickets'(T, _, _, _, _, _, _, C, _).
origin(T, O) ← cheaptickets'(T, _, _, _, _, _, O, _, _).
destination(T, A) ← cheaptickets'(T, _, _, _, _, _, _, _, A).
return(T, RD) ← cheaptickets'(T, _, _, _, _, RD, _, _, _).
depart(T, DD) ← cheaptickets'(T, _, _, _, DD, _, _, _, _).
hasFare(T, F)← cheaptickets'(T, _, F, _, _, _, _, _, _).
hasTax(T, Tx) ← cheaptickets'(T, _, _, Tx, _, _, _, _, _).
transitFee(T, F) ← cheaptickets'(T, _, _, _, _, _, _, C, _), context(receiver, c_ben), value(C, c_ct, "United Kingdom"), value(F,c_ct,27).

transitFee(T,F) ← cheaptickets'(T, _, _, _, _, _, _, C, _), value(F,_,0).
shippingCost(T, S)← cheaptickets'(T, _, _, _, _, _, _, _, _), value(S,c_ct,20).
serviceFee(T, SF) ← cheaptickets'(T, _, _, _, _, _, _, _, _), value(SF,c_ct,5).

CONTEXT ONTOLOGY

Modifiers:

lformat(airport, Context, airportFormat), dformat(date, Context, dateFormat),
currency(moneyAmt, Context, currencyType), inclusion(airfare, Context, inclusionType),
coverage(airfare, Context, coverageType).

CONTEXTS

Ben

currency(F, c_ben, M) ← M = object (currencyType, "USD", c_ben, _).
inclusion(F, c_ben, M) ← M = object(inclusionType, "+shipping" , c_ben, _).
coverage(F, c_ben, M) ← M = object (coverageType, "round-trip", c_ben, _).
lformat(A, c_ben, M) ← M = object (airportFormat, "city_name", c_ben,_).
dformat(D, c_ben, M) ← M = object (dateFormat, "dd/mm/yy", c_ben, _).

Cheaptickets

currency(F, c_ct, M) ← M = object (currencyType, "USD", c_ct, _).
inclusion(F, c_ct, M) ← M = object(inclusionType, "nominal" , c_ct, _).
coverage(F, c_ct, M) ← M = object (coverageType, "one-way", c_ct, _).
lformat(A, c_ct, M) ← M = object (airportFormat, "airport_code", c_ct, _).
dformat(D, c_ct, M) ← M = object (dateFormat, "mm/dd/yy", c_ct, _).
currency(F, c_ct, M) ← transitFee(T,F), cxnCountry(T,C), countryCurrency(C, M).

Ancillary Sources

dformat(D, c_ancillary, M) ← M = object (dateFormat, "mm/dd/yy", c_ancillary, _).

CONVERSION FUNCTION NETWORK

Finclusion: Bi-directional, Piece-wise, Priority: 100
Finclusion(X, VX, SC, "nominal", "+tax", TC, VY) ←
hasFare(T, X), hasTax(T, Tx), value(Tx, SC, VTx),sum(VX, VTx, VY).
Finclusion(X, VX, SC, "+tax", "+transit", TC, VY) ←
hasFare(T, X), transitFee(T, Tf), value(Tf, SC, VTf),sum(VX, VTf, VY).
Priority: 300
Finclusion(X, VX, SC, "+transit", "+service", TC, VY) ←
hasFare(T, X), serviceFee (T, S), value(S, SC, VS), sum(VX, VS, VY).
Finclusion(X, VX, SC, "+service", "+shipping", TC, VY) ←hasFare(T, X), shippingFee(T, S), value(S, SC, VS),sum(VX, VS, VY).
Fcoverage: Bi-directional, Priority: 200
Fcoverage(X, VX, SC, "one-way", "round-trip", TC, VY) ←mul(VX,2,VY).
Mcurrency: Bi-directional, Priority: 400
Mcurrency(X, VX, SC, VCurrencyX, VCurrencyY, TC, VY) ←
value(Today, SC, VToday), system_date(VToday), value(CurrencyX, SC, VCurrencyX), value(CurrencyY, SC, VCurrencyY), currencyrates'(CurrencyX, CurrencyY, Today, Rate),
value(Rate, SC, VRate), mul(VX, VRate, VY).
Ddformat: Bi-directional, Priority: 100
Ddformat(X, VX, SC, "dd/mm/yy", "mm/dd/yy", TC, VY) ← dateConverter(X,Y), value(Y, TC, VY).
Alformat: Bi-directional, Priority: 100
Alformat(X, VX, SC, "city_name", "airport_code", TC, VY) ← cityairport'(X,Y), value(Y, TC, VY).

Note that in this case priority numbers are assigned to each mapping, because the modifiers of the Airfare type are non-orthogonal (i.e. cannot be applied in any order). For example, performing mappings for the inclusion modifier before and after the coverage modifier produces two different results. If the coverage modifier is applied

first when going from {nominal, one-way, USD} to {+tax, round-trip, USD} we get 2VX + VT; and if the inclusion modifier is applied first we get 2VX+ 2VT. When there are non-orthogonal mappings, priorities are needed to determine the correct order of applying the conversions.
System_date is a system function, and dateConverter is a simple program that does date parsing.

APPENDIX B

GENERIC SYMBOLIC EQUATION SOLVING CONSTRAINTS IN ECOIN (SAMPLE)

Constraint Set 1: Ground variables are bound

Examples:

sum(X,Y,Z) ==> ground(X) | bound(X).
sum(X,Y,Z) ==> ground(Y) | bound(Y).
sum(X,Y,Z) ==> ground(Z) | bound(Z).

Constraint Set 2: Variables functionally determined by ground variables are ground & bound. Their values can be calculated immediately.

Examples:

div(X,Y,Z) <=> ground(X), ground(Y), nonground(Z), Y ~=0 | Z is X / Y, bound(Z).
div(X,Y,Z) <=> ground(X), ground(Z), nonground(Y), Z ~=0 , X ~=0 | Y is Z * X, bound(Y).
div(X,Y,Z) <=> ground(Y), ground(Z), nonground(X),Y ~=0 | X is Z * Y, bound(X).
div(0,Y,Z) <=> nonground(Z), Y ~=0 | Z is 0, bound(Z).
div(X,Y,0) <=> nonground(X), Y ~=0 | X is 0, bound(X).
div(X,0,Z) <=> false.

Constraint Set 3: Result variables functionally determined by bound values are bound

Example:

sum(X,Y,Z), bound(X), bound(Y) ==> bound(Z).

Constraint Set 4: Identity element constraints

Examples:

mul(1,Y,Z) <=> nonground(Z), nonground(Y) | Z=Y.
mul(X,1,Z) <=> nonground(Z), nonground(X) | Z=X.

Constraint Set 5: Arithmetic integrity constraints

Examples:

sub(A,Y,Z), sub(X,Y,Z) ==> nonground(X), nonground(A) | X = A.
sub(X,Y,Z), sub(X,A,Z) ==> nonground(Y), nonground(A) | Y = A.
sub(X,Y,A), sub(X,Y,Z) ==> nonground(Z), nonground(A) | Z = A.

Constraint Set 6: Predicates with bound result variables are simplified to their inverses

Examples:

sum(X,Y,Z), bound(Z) <=> sub(Z,Y,X), bound(Z).
sub(X,Y,Z), bound(Z) <=> sum(Y,Z,X), bound(Z).
mul(X,Y,Z), bound(Z) <=> Y~=0 | div(Z,Y,X), bound(Z).
div(X,Y,Z), bound(Z) <=> Y~=0 | mul(Y,Z,X), bound(Z).
pow(X,Y,Z), bound(Z) <=> Y~=0 | pow(Z,N,X), div(1,Y,N), bound(N).

Constraint Set 7: Predicates with a bound result variable and another bound variable are simplified with their inverses while declaring the remaining variable bound.

Examples:

sum(X,Y,Z), bound(X), bound(Z) <=> sub(Z,X,Y), bound(X), bound(Y), bound(Z).
sum(X,Y,Z), bound(Y), bound(Z) <=> sub(Z,Y,X), bound(X), bound(Y), bound(Z).

Constraint Set 8: Interaction constraints

Examples:

mul(X,A,Y), sub(B,Y,X) <=> A~=-1 | div(B,N,X), sum(1,A,N).
mul(X,A,Y), sum(B,Y,X) <=> A~=-1 | div(B,N,X), sub(1,A,N).
div(X,A,Y), sub(B,Y,X) <=> A~=-1 | mul(A,B,N1), sum(1,A,N2), div(N1,N2,X).
div(X,A,Y), sum(B,Y,X) <=> A~=-1 | mul(A,B,N1), sub(1,A,N2), div(N1,N2,X).

Constraint Set 9: Miscellaneous Simplification Constraints

Example:
mul(X1,C,Z1), mul(X2,C,Z2) <=> sum(X1, X2, X), sum(Z1, Z2, Z), mul(X,C,Z).

APPENDIX C

Sample Activation Of Symbolic Equation Solving Constraints

After the abducibles are obtained, the abduction phase ends, and the constraint satisfaction phase starts. At this stage, the abducibles, which are mostly in the non-ground form ∃X (A(X), C(X)), are further processed by the symbolic equation solving constraints. During this process, arithmetic predicates are inverted, combined and simplified into new forms. To illustrate this process, we consider the first set of abducibles from the original query, together with the symbolic equation solving rules. The following arithmetic predicates are the first set of abducibles:

{sum(Airfare, T, PT), mul(ExchangeRate,27,VFA),
sum(PT,VFA,PTV), sub(PTVS,5,PTV),
sub(Final, 20, PTVS), sum(20,5,SP),
sub(Final, SP, FSP), div(RT,2,FSP),
sum(RT,SP,VAirfare). }

In processing the above set of constraints, the following CHR rules are activated:
Constraint Set 1: Ground variables are bound
sub(Final,20,PTVS) ==> ground(20) | bound(20).
sub(PTVS,5,PTV) ==> ground(5) | bound(5).
div(RT,2,FWOP) ==> ground(2) | bound(2).
mul(ExchangeRate,27,VFA) ==> ground(27) | bound(27).
Constraint Set 2: Variables functionally determined by ground variables are ground & bound. Their values can be calculated immediately.
sum(20,5,SP), bound(20), bound(5) <=> SP is 25.
Constraint Set 3: Result variables functionally determined by bound values are bound
sum(Airfare, T, PT), bound(Airfare), bound(T) ==> bound(PT).
T and Airfare are from the cheaptickets relation; therefore are implicitly bound.
mul(ExchangeRate,27,VFA), bound(ExchangeRate), bound(27) ==> bound(VFA).
ExchangeRate is from the currencyrates relation; therefore is implicitly bound.
sum(PT,VFA,PTV), bound(PT), bound(VFA) ==> bound(PTV).
Constraint Set 6: Predicates with bound result variables are simplified with their inverses
sub(PTVS,5,PTV), bound(PTV) <=> sum(PTV,5,PTVS), bound(PTVS).
sub(Final, 20, PTVS), bound(PTVS) <=> sum(PTVS,20,Final), bound(Final).
Constraint Set 3: Result variables functionally determined by bound values are bound
sub(Final, 25, FSP), bound(Final), bound(25) ==> bound(FSP).
Constraint Set 6: Predicates with bound result variables are simplified with their inverses
div(RT,2,FSP), bound(FSP) <=> mul(FSP,2,RT), bound(RT).
Constraint Set 3: Result variables functionally determined by bound values are bound
sum(RT,25,VAirfare), bound(RT), bound(25) ==> bound(VAirfare).
At this stage the following set of arithmetic predicates remain in the constraint store:
{ sum(Airfare, T, PT), mul(ExchangeRate,27,VFA), sum(PT,VFA,PTV),
sum(PTV,5,PTVS), sub(PTVS, 20, Final),
sub(Final, 25, FSP), mul(FSP,2,RT), sum(RT,25,VAirfare)}
These are now used to construct:
VAirfare = (Airfare +Tax + ExchangeRate* 27 + 5 + 20 -25) * 2 + 25
Further simplifications are done by using additional simplification constraints. For example the following propagation is obtained

sum(PTV,5,PTVS), sum(PTVS, 20, Final) ==> sum(PTV,25,Final).
by using:
Constraint Set 9. Miscellaneous Simplification Constraints
sum(X,Y,Z),sum(Z,A,B) ==> ground(Y),ground(A),nonground(X),nonground(B) | C is Y+A, sum(X, C, B).
Furthermore the following simplification is performed:
sum(PTV,25,Final), sub(Final, 25, FSP) <=> FSP=PTV, sum(PTV,25,Final).
by using
Constraint Set 5: Arithmetic integrity constraints
sum(X,A,Y), sub(Y, A, Z) <=> X=Z, sum(X,A,Y) or
sum(X,A,Y), sub(Y, A, Z) <=> X=Z, sub(Y,A,X)
Finally, the program terminates with the following set of abducibles:
{answer(VAirline,VAirfare),
  cheaptickets(I,VAirline, Airfare, T, “06/01/07”, “07/01/07”, Airport1, “United Kingdom”, Airport2),
  cityairport(“Boston”, Airport1), cityairport(“Istanbul”, Airport2),
  currencyrates(“GBP”,”USD”, ExchangeRate, “05/01/07”),
  sum(Airfare, T, PT), mul(ExchangeRate,27,VFA),
  sum(PT,VFA,PTV), mul(PTV,2,RT), sum(RT,25,VAirfare).
}
and the next set is obtained similarly
{answer(VAirline,VAirfare),
cheaptickets(I,VAirline,Airfare,T,“06/01/07”,“07/01/07”,Airport1, Cxn, Airport2), Cxn <> “United Kingdom”,
cityAirport(“Boston”, Airport1), cityAirport(“Istanbul”, Airport2),
sum(Airfare, T, PT), mul(PT,2,RT), sum(RT,25,VAirfare).}
Together these trivially translate to the final mediated query MQ1 shown in Section 2.