Data Semantics: Data Integration and the Semantic Web

Yuan An
Department of Computer Science
University of Toronto
yuana@cs.toronto.edu

Document prepared for the Depth Oral

In BA 7256, At 2:00-4:00pm, Thursday, January 22nd, 2004

Abstract

Information systems today need to resolve the heterogeneity among data residing in multiple autonomous data sources. In particular, the use of the World Wide Web as a universal medium for exchanging information has radically transformed our vision about data access and manipulation. Following the perspective of the semantic web, we believe that the explicit presentation of data semantics will facilitate data interoperation in a variety of data manipulation tasks.

Investigating and capturing the meaning of data is a core problem in all of applications in computer science. Especially, in database area the problem has been studied by many researchers. As a result, we have seen much progress on topics such as *schema integration, schema matching, schema mapping, and generic model management* in multidatabases, federated databases, and data integration over past two decades. The solutions that have been developed so far are a mixture of semi-automatic, heuristic-driven, and multi-layered solutions.

Representing and reasoning about human knowledge in terms of symbolic structures is another long-standing problem. Consequently, many logical formalisms have been proposed bearing distinct expressivity and reasoning capabilities. A recent application of this work called *semantic web* aims at providing a machine-understandable Web by organizing data in terms of their semantics. It employs the approach of annotating data with formal ontologies. This approach demonstrates one aspect of the paradigm of ontology-based information representation and integration.

We believe that investigating data semantics as to establish and maintain the correspondence between data/model and its intended subject matter could have long-term benefits as well as instant gains. Our future line of research is founded on a principle of semantic correspondence continuum. The continuum is grounded on domain ontologies.

1 Introduction

1.1 Data Semantics as an Enduring Problem

Information systems today manipulate data originating at multiple autonomous and heterogeneous data sources. Consequently, state-of-the-art data management systems are facing the paradigm shift from a monolithic and controllable environment to a distributed and open-ended one. In particular,

resolving semantic heterogeneity, and gathering and sharing data among autonomous and heterogeneous data sources are key to the new paradigm. Some current applications include data warehousing that supports decision by providing historical, summarized, and consolidated data from multiple operational databases; data integration systems that answer queries posed against a global schema by using disparate local data sources; e-business applications that enable service providers and consumers to locate partners and fulfill transactions automatically; web portals that integrate information from different sources; and the vision of the semantic web that aims at the creation of machine-understandable Web data so that humans and computers can interact freely to satisfy the ever-growing need for swift information consumption. Although various approaches to resolving semantic heterogeneity have been developed, the profound and elusive questions yet remain [She97]: What is data semantics? Where does data semantics come from? and How to represent data semantics?

1.2 An Overview of Data Integration and the Semantic Web

The prevalence of autonomy, heterogeneity, connectivity, openness, and reuseness in the modern information systems environment calls for interoperability. Two lines of research give insights into the investigation of data semantics to promote interoperability: data integration and the semantic web – the recent descendant of knowledge representation.

Data integration is the problem of combining data residing in different sources, and providing the user with a unified view, global schema, of these data. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the relation between the sources and the global schema is therefore a crucial aspect. It calls for mappings.

A mapping is an assertion about query expressions of source schemas and the global schema. Two basic approaches have been proposed to this purpose. The first approach, called *global-as-view* (GAV), requires that the global schema is expressed in terms of the sources. The second approach, called *local-as-view* (LAV), requires the global schema to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema. The processing of queries posed in terms of the global schema in data integration requires a query reformulation. Techniques like *answering query using views* and *rewriting query using views* are tightly bound to the LAV approach.

A great deal of effort has been put into the study of increasing the degree of automation of mapping generation. Familiar topics are *schema integration*, *schema matching*, *schema mapping*, and *generic model management*, to name a few. All of them are valuable approaches in dealing with semantic heterogeneity. We will discuss several typical approaches in corresponding sections later on.

On the other hand, the semantic web aims at directly providing machine-understandable data on the Web. It has two salient ingredients: (i) web ontologies; (ii) data annotation. Although it has long way to go for the realization of the semantic web involving the development of ontology, language, inference engine, and other components, the principle of the semantic web reminds us that data semantics is to establish and maintain the correspondence from data to the subject matter it intended.

Along the line of knowledge representation, many specific logical formalisms have been proposed for overcoming the undecidability of the full first order logic. Even though having high

order computational complexity, Description Logics enjoys the decidability of concept satisfiability and subsumption, and has been adopted into many knowledge representation cases. Specifically, Description Logics is widely used in representing ontology. Philosophically, an ontology is a systematic account of existence. In computer science, an ontology is an explicit specification of conceptualization for a subject matter. It can be seen that the paradigm of ontology-based information integration has gained increasing attention, theoretically and practically.

The commonality of data integration and the semantic web is to overcome semantic heterogeneity among interconnected data sources. Although there are data integration systems using ontology as information broker, approaches of data integration tend to focus on internal structures and heuristic manners, partly because of keeping the efficiency of processing large amount of data. In contrast, the semantic web approach turns to explicit representation of data semantics in data sources, but it is still in an infant stage. In sum, creating machine-understandable data remains the holy grail of data integration as well as of the semantic web.

1.3 Continuing to Chase the Holy Grail

The fundamental problem we are faced with is that data are always expressible and interpretable in multiple different ways. One reason is the model heterogeneity. Models are the outcomes of a modelling process which describes a complex phenomenon by relating it with another familiar set of structures. Database schema is a type of model, and people have developed various, heterogeneous types of data modelling languages for representing the same real world states. The other reason is the lack of maintenance of the correspondence between data/model and its intended subject matter. We need ways to disambiguate data for automatic interoperation.

In human communication, participants employ their common background knowledge to make natural language meaningful in their minds. Likewise, in machine communication, both parties engaged in a communication should have common knowledge as well. Knowledge could be manifested as symbolic structures such as formal ontologies and correspondence formulas. In short, meaning of data should be presented explicitly and accurately and should be machine-understandable. In what follows, we examine efforts in the field of data integration and the semantic web, and we propose one step in the direction of explicitly representing semantics towards automatic data interoperation for a variety of applications.

1.4 Document Structure

The rest of the document is organized as follows: Section 2 presents a common database design scenario to elaborate the prevalence of the problem of data semantics. Section 3 discusses data integration theoretically and practically. Section 4 studies schema manipulation endeavors in database field. Section 5 investigates the principles of the semantic web. Section 6 shows the development of knowledge representation techniques and ontology. Section 7 proposes the future research direction. Finally, Section 8 gives a few concluding remarks.

2 Semantics in a Common Database Design and Integration Scenario

Database research and practices have provided a rich body of scenarios and examples for the investigation of data semantics and interoperability. In this section, we describe a common database

design and integration scenario. In which we argue that correspondence implies semantics.

Consider two universities, *UofT* and *UofW*, designing their student information systems, respectively. Relational databases will be used to store operational data and XML files will be used to publish data on the Web. Usually, the procedure is as follows: (i) database designers establish conceptual schemas (e.g., Entity-Relationship schemas) by modelling the states of affairs of the real world; (ii) by some standard procedures, they translate the conceptual schemas into relational schemas which are used to store the operational data; (iii) designers publish XML data on the Web, in which the data are based on the relational database backend. Such a general procedure for information system design is shown in Figure 1.

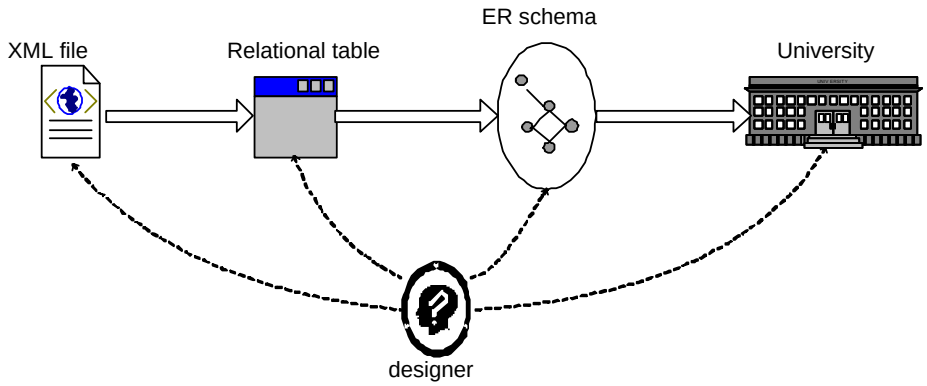

Figure 1: A university information systems design procedure

The database designers of *UofT* produce an Entity-Relationship schema for modelling their subject matter. Let $\mathcal{E}_{ut}$ denote the ER schema. A portion of $\mathcal{E}_{ut}$ is shown in Figure 2. The ER schema $\mathcal{E}_{ut}$ consists of elements such as entities, relationships, and attributes. Each element means that it is the representation of the real world counterpart. For example, the *student* entity represents the student objects in the university of *UofT*. From the extensional point of view, the content of *student* database coincides the values or tuples of values for representing the students in *UofT*, nothing else. However, there is a conceptual sense carried by the *student* entity. That is, the concept of university student who is a person studying in an academic organization. We will distinguish the conceptual sense of a modeling element from its extensional definition. We call it intensional definition, and we will make use of it in the representation of data semantics.

For real data storage and management, the conceptual schema needs to be translated into a logical schema such as a relational schema. Let $\mathcal{R}_{ut}$ denote the relational schema derived from $\mathcal{E}_{ut}$. Several relations are shown in Figure 3. Apparently, there is a representational relationship from the relational schema $\mathcal{R}_{ut}$ to the ER schema $\mathcal{E}_{ut}$.

Finally, $\mathcal{X}_{ut}$ denotes the XML schema shown in Figure 4 which is used for describing the information published on the Web. The information is based on the data stored under the relational schema $\mathcal{R}_{ut}$. Again, we can say that the XML schema $\mathcal{X}_{ut}$ represents the relational schema $\mathcal{R}_{ut}$ in a different structure.

In the same way, $\mathcal{E}_{uw}$ shown in Figure 5 denotes the ER schema of UW, and $\mathcal{R}_{uw}$ shown in Figure 6 denotes its relational schema and $\mathcal{X}_{uw}$ denotes its XML schema in Figure 7. Consequently, we can characterize the corresponding relationships between schemas both intensionally and exten-

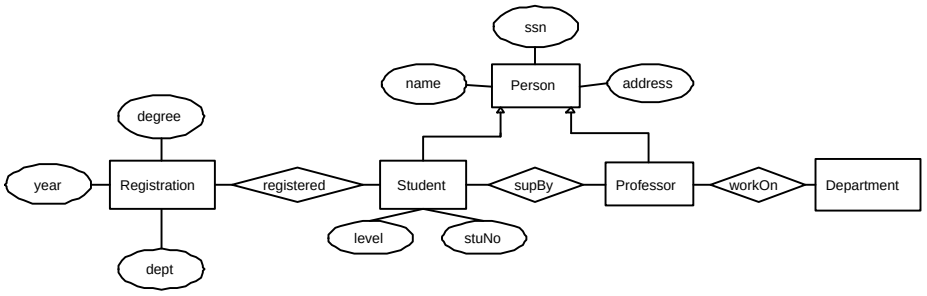

Figure 2: $\mathcal{E}_{ut}$: the Entity-Relationship diagram of the information systems of UT

```
Student(stuNo, Name, Address, level, SupBy)
Registration(stuNo, Year, Degree, Dept)
```

Figure 3: $\mathcal{R}_{ut}$: the relational schema of the information systems of UT

sionally. We set up the characterization as part of the goal of this paper.

```
UTstudentDB:
 Students:  set of
   Student
      @level
      StuNo
      Name
      Address
    registrations:  set of
      Registration
            Degree
            Department
            Year
    Professors:  set of
      SupBy
          @name
```

Figure 4: $\mathcal{X}_{ut}$: the XML schema of the Web-based information systems of UT

Questions revolving around these schemas in terms of data interoperation/integration can be raised as follows:

Question1: Finding the semantical mapping, $\mathcal{R}_{uw} \rightarrow \mathcal{R}_{ut}$, from schema $\mathcal{R}_{uw}$ to $\mathcal{R}_{ut}$ to merge data from *UofW* onto *UofT*, or vice versa.

Question2: Finding the semantical mapping, $\mathcal{X}_{uw} \rightarrow \mathcal{X}_{ut}$, from schema $\mathcal{X}_{uw}$ to $\mathcal{X}_{ut}$ to merge, translate, and integrate data on the website of *UW* onto the data on the website of *UofT*, or vice versa.

Question3: Finding the semantical mapping from the relational schema of one university to the XML schema of another university, or vice versa.

Questions like these have been studied intensively and have drawn several lines of research in the management of semantic heterogeneity. The purpose and the meaning of the semantical mapping

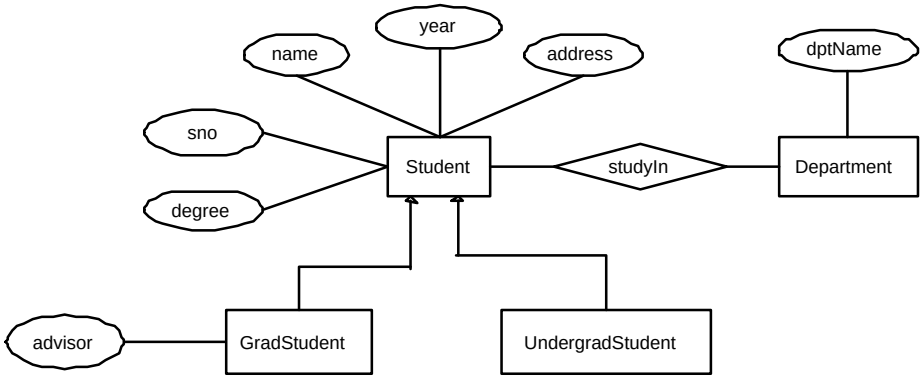

Figure 5: $\mathcal{E}_{uw}$: the Entity-Relationship diagram of the information systems of UW

```
GradStudent(sno, name, address, year, degree, advisor)
UndergradStudent(sno, name, address, year, degree)
StudyIn(sno, dptName)
```

Figure 6: $\mathcal{R}_{uw}$: the relational schema of the information systems of UW

```
UWstudentDB:
    UndergradStudents:  set of
      UndergradStudent
         Sno
         Name
         Address
         Year
         Department
         Degree
    GradStudents:  set of
      GradStudent
         Sno
         Name
         Address
         Year
         Department
         Degree
       advisors:  set of
         Advisor
           name
           Department
```

Figure 7: $\mathcal{X}_{uw}$: the XML schema of the Web-based information systems of UW

aside, there is a clear distinction between the lineal vertical correspondence of schemas, which is derived from the modelling process in the designer's mind, and the horizontal semantical mapping, which is generated by tools or user between arbitrary schemas with overlap as shown in Figure 8.

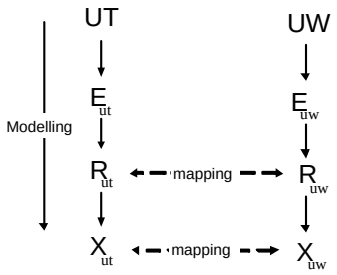

Figure 8: Lineal correspondence vs. semantic mapping.

Unfortunately, the vertical correspondences/relations between pair models/schemas are discarded as soon as schemas needed have been derived. Consequently, in database research, a great deal of attention has been directed into the development of various techniques of *normalization* concerning some criteria such as *information lossless* and *redundancy reduction*. Data semantics is merely presented through sets of *integrity constraints*, saying nothing about its correspondences to real world objects. Information integration among multiple data sets is solved as long as the meaning of data is entrusted to a small group of users and/or application programs. However, the terrain of information integration has been expanded as wide as World Wide Web. We believe that keeping the lineal correspondences between schemas for an arbitrary intelligent agent to process could be an efficient and effective way of information gathering and sharing in a global scope. Thus it calls for the formalism of the real world objects and a formal correspondence representation.

To complete the scenario, we assume the existence of a formalized domain ontology which serves the role of the states of affairs of the real world. Let $\mathcal{O}$ denote the ontology. Such a ontology prescribes common concepts, properties of a concept, and relationships among the concepts for a class of academic organizations including universities. In a Description Logic [CLN98], the portion of the ontology $\mathcal{O}$ is shown in Figure 9. The ontology described in a DL can be easily translated into the description of the OWL which is the on-going standardization of web ontology language.

By virtue of the ontology, we are equipped with the necessary formalism for the explicit specification of data semantics in terms of correspondence. The rest of the work is to investigate efforts from both data integration and the semantic web, and to propose the future research direction.

3 Data Integration Theory and Application

3.1 Data Integration Theory

3.1.1 Formalization of Data Integration Problem

A data integration system $\mathcal{I}$ is formalized [Len02] in terms of a triple $< \mathcal{G}, \mathcal{S}, \mathcal{M} >$, where

- $\mathcal{G}$ is a global schema expressed in a language $\mathcal{L}_{\mathcal{G}}$ over an alphabet $\mathcal{A}_{\mathcal{G}}$.

```
PERSON ⊑ LIVINGTHING⊓∀hasSSN.STRING⊓=1hasSSN⊓∀hasAddress.ADDRESS
⊓∀hasName.PERSON-NAME⊓∃=1hasName.
PERSON-NAME⊑∀hasFirstName.STRING⊓∀hasLastName.STRING.
UNIVERSITY ⊑ ORGANIZATION⊓∀hasAddress.ADDRESS⊓∀hasName.ORG-NAME.
ORG-NAME ⊑ STRING.
UNIVERSITY-STUDENT ⊑ PERSON⊓∀hasStuno.STRING⊓∃=1hasStuno⊓
∀registerAt.REGISTRATION⊓∃≥1registerAt.
REGISTRATION ⊑ ∀ofUniv.UNIVERSITY⊓∀inYear.YEAR⊓∀withDegree.DEGREE⊓
∀ofDept.DEPARTMENT
DEPARTMENT ⊑ ORGANIZATION⊓∀hasName.ORG-NAME⊓∀affiliatedWith.UNIVERSITY.
GRAD-DEGREE ⊑ DEGREE⊓∀hasName.{PhD,MSc,MEng,MA}.
UNDERGRAD-DEGREE ⊑ DEGREE⊓∀hasName.{BSc,BEng,BA}.
GRAD-REGISTRATION ⊑ REGISTRATION⊓∀withDegree.GRAD-DEGREE.
UNDERGRAD-REGISTRATION ⊑ REGISTRATION⊓∀withDegree.UNDERGRAD-DEGREE.
GRAD ⊑ UNIVERSITY-STUDENT⊓∃registerAt.(∀withDegree.GRAD-DEGREE)⊓
∀hasAdvisor.PROFESSOR⊓∃≥1hasAdvisor.
UNDERGRAD ⊑ UNIVERSITY-STUDENT⊓∀registerAt.(∀withDegree.UNDERGRAD-DEGREE).
```

Figure 9: $\mathcal{O}$: an ontology of academic organizations

- $\mathcal{S}$ is a source schema expressed in a language $\mathcal{L}_\mathcal{S}$ over an alphabet $\mathcal{A}_\mathcal{S}$.
- $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of assertions of the forms $q_\mathcal{S} \rightsquigarrow q_\mathcal{G}$ or $q_\mathcal{G} \rightsquigarrow q_\mathcal{S}$ ($q_\mathcal{G}$ and $q_\mathcal{S}$ are queries over $\mathcal{G}$ and $\mathcal{S}$, respectively).

The data integration system $\mathcal{I} =< \mathcal{G}, \mathcal{S}, \mathcal{M} >$ aims at answering a query q posed against the global schema $\mathcal{G}$ by data stored in the source $\mathcal{S}$. The semantics of the data integration system is specified by *certain answers* described as follows:

> Given a source database, $\mathcal{D}$, for $\mathcal{I}$ over a fixed domain Γ, i.e., $\mathcal{D}$ conforms to the source schema $\mathcal{S}$ and satisfies all constraints in $\mathcal{S}$. A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be *a legal database* of $\mathcal{I}$ with respect to $\mathcal{D}$, if:
>
> - $\mathcal{B}$ is legal with respect to the global schema $\mathcal{G}$, i.e., $\mathcal{B}$ conforms to $\mathcal{G}$ and satisfies all constraints of $\mathcal{G}$;
> - $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ specified as follows:
>   - If source $\mathcal{S}$ is sound, then $q_\mathcal{S}^\mathcal{D} \subseteq q_\mathcal{G}^\mathcal{B}$;
>   - If source $\mathcal{S}$ is complete, then $q_\mathcal{S}^\mathcal{D} \supseteq q_\mathcal{G}^\mathcal{B}$;
>   - If source $\mathcal{S}$ is exact, then $q_\mathcal{S}^\mathcal{D} = q_\mathcal{G}^\mathcal{B}$.
>
> (Given a query q posed against the global schema $\mathcal{G}$. We denote with $q^{\mathcal{DB}}$ the set of tuples in a database $\mathcal{DB}$ that satisfy q.) The answer $q^{\mathcal{I},\mathcal{D}}$ to the query q in the data integration system $\mathcal{I}$ with respect to $\mathcal{D}$, is the set of tuples t of objects in Γ such that $t \in q^\mathcal{B}$ for *every* global database $\mathcal{B}$ that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$. The set $q^{\mathcal{I},\mathcal{D}}$ is called the set of *certain answers* to q in $\mathcal{I}$ with respect to $\mathcal{D}$.

Two basic approaches for specifying the mapping in a data integration system have been proposed in literature, called *local-as-view (LAV)*, and *global-as-view (GAV)*, respectively. A LAV mapping is a set of assertions, one for each element s of $\mathcal{S}$ of the form

$$s \rightsquigarrow q_\mathcal{G}$$

From the modelling perspective, the LAV approach is based on the idea that the content of each source s should be characterized in terms of a view $q_{\mathcal{G}}$ over the global schema.

On the other hand, the mapping $\mathcal{M}$ of the GAV approach associates each element g in $\mathcal{G}$ with a query $q_{\mathcal{S}}$ over $\mathcal{S}$. Therefore, a GAV mapping is a set of assertions, one for each element g of $\mathcal{G}$, of the form

$$g \rightsquigarrow q_{\mathcal{S}}$$

From the modelling perspective, the GAV approach is based on the idea that the content of each element g of the global schema should be characterized in terms of a view $q_{\mathcal{S}}$ over the source.

Note that the LAV approach favors the extensibility of the system. In other words, adding a new source simply means enriching the mapping with a new assertion, without other changes. In contrast, in principle, the GAV approach favors the system in carrying out query processing, because it tells the system how to use the sources to retrieve data.

In addition to LAV and GAV, the *global-local-as-view* (GLAV) approach is also introduced in literature. In GLAV, the relationships between the global schema and the sources are established by making use of both LAV and GAV assertions. More precisely, in a GLAV mapping, every assertion has the form $q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}}$, where $q_{\mathcal{S}}$ is a query expression over the source schema $\mathcal{S}$, and $q_{\mathcal{G}}$ is a query expression over the global schema $\mathcal{G}$.

3.1.2 View-based Query Answering

Query processing in a data integration system involves different manners in terms of LAV or GAV modelling approaches. In LAV, the problem of processing a query is traditionally called *view-based query processing* which is classified into *view-based query rewriting* and *view-based query answering*. The query answering is exactly the problem of computing the certain answers to q with respect to a source database. Computing certain answers has been extensively investigated in literature. [AD98] carries out a comprehensive analysis of the complexity of the problem under the different assumptions where the views and the queries are expressed in terms of various languages: conjunctive queries without and with inequalities, positive queries, Datalog, and first-order queries. The complexity is measured with respect to the size of the view extensions (data complexity). Table 1 shows the data complexity of the problem under the open world assumption. Table 2 shows the data complexity of the problem under the closed world assumption. Note that, for the query languages considered, the closed world assumption, i.e., the exact view, complicates the problem.

	query				
views	CQ	$CQ^{\neq}$	PQ	$Datalog$	FO
CQ	PTIME	co-NP	PTIME	PTIME	undecidable
$CQ^{\neq}$	PTIME	co-NP	PTIME	PTIME	undecidable
PQ	co-NP	co-NP	co-NP	co-NP	undecidable
$Datalog$	co-NP	undec.	co-NP	undec.	undecidable
FO	undec.	undec.	undec.	undec.	undecidable

Table 1: Data complexity of query answering under open world assumption.

	query				
views	CQ	$CQ^{\neq}$	PQ	$Datalog$	FO
CQ	co-NP	co-NP	co-NP	co-NP	undecidable
$CQ^{\neq}$	co-NP	co-NP	co-NP	co-NP	undecidable
PQ	co-NP	co-NP	co-NP	co-NP	undecidable
$Datalog$	undec.	undec.	undec.	undec.	undecidable
FO	undec.	undec.	undec.	undec.	undecidable

Table 2: Data complexity of query answering under closed world assumption.

3.1.3 View-based Query Rewriting

Query rewriting aims at reformulating, in a way that is independent from the current source databases, the original query in terms of a query to the sources. Apparently, it may happen that no rewriting in the fixed target query language $\mathcal{L}_{\mathcal{Q}}$ exists that is *equivalent* to the original query. In this case, people are interested in computing a so-called *maximally containment rewriting*. The comprehensive survey [Hal01] discusses the large body of work on incorporating materialized views into query optimizers, and specific algorithms for query rewritings.

The definition of two types of query rewriting: *equivalent rewritings* and *maximally-contained rewritings* are given below.

Definition (Equivalent rewritings). Let Q be a query and $\mathcal{V} = V_1, ..., V_m$ be a set of view definitions. The query Q' is an equivalent rewriting of Q using $\mathcal{V}$ if:

- the subgoals of Q' are either relations in $\mathcal{V}$, or comparison predicates, and
- Q' is equivalent to Q.

□

Definition (Maximally-contained rewritings). Let Q be a query, $\mathcal{V} = V_1, ..., V_m$ be a set of view definitions, and $\mathcal{L}$ be a query language. The query Q' is a maximally-contained rewriting of Q using $\mathcal{V}$ w.r.t $\mathcal{L}$ if:

- Q' is a query in $\mathcal{L}$ that refers to the views in $\mathcal{V}$ or comparison predicates,
- Q' is contained in Q, and
- there is no rewriting $Q_1 \in \mathcal{L}$, such that $Q' \subseteq Q_1 \subseteq Q$ and Q_1 is not equivalent to Q'.

□

When developing algorithms that produce rewritten queries, one can ask two questions [Hal00]: (i) whether the algorithms is sound and complete: given a query Q and a set of views $\mathcal{V}$, is there an algorithm that will find a rewriting of Q using $\mathcal{V}$ when one exists; (ii) what is the complexity of that problem. For the class of queries and views expressed as conjunctive queries, [LMSS95] shows that when the query does not contain comparison predicates and has n subgoals, then there exists an equivalent conjunctive rewriting of Q using $\mathcal{V}$ only if there is a rewriting with at most n subgoals.

An immediate corollary is that the problem of finding an equivalent rewriting of a query using a set of views is in NP, because it suffices to guess a rewriting and check its correctness. [Hal00] also points out that the problem of finding a contained rewriting is NP-complete.

The bound on the size of the rewriting (and therefore on the search space of rewritings) has led to a succession of algorithms that attempt to efficiently search the space. [Hal01] gives three of these algorithms: the Bucket Algorithm, the Inverse-rules Algorithm, and the MiniCon Algorithm.

As we can see, the maximally-contained query rewriting needs to check the query containment. Here, we refer to the literature containing most of the results on query containment. [CM77] establishes NP-completeness for conjunctive queries. [Klu88, vdM92] proves Π_2^p- completeness of containment of conjunctive queries with inequalities. [SY80] studies the case of queries with the union and difference operators. [CV92, vdM92] presents results of the decidability and undecidability of various classes of Datalog queries with inequalities.

3.1.4 Query Processing in GAV Approach

Query processing through the GAV approach can be based on a simple unfolding strategy if the integrity constraints in the global schema is absent and the views are exact. However, when the global schema allows integrity constraints, and the views are sound, then query processing in GAV systems becomes more complex. [CCGL02] shows that the simple unfolding algorithm does not retrieve all certain answers in the presence of integrity constraints of the global schema. The correct abstract representation of the set of legal global databases is termed as *canonical database* in which new tuples complying with the integrity constraints are inserted into the "*retrieved global database*" which is populated by the source according to the mapping.

In order to compute the all certain answers without computing the infinite canonical database, [CCGL02] presents a sound and complete algorithm through the partial evaluation of logic programming in the case that the language for expressing both the user query and the queries in the mapping is the one of union of conjunctive queries. To process a query q, [CCGL02] expands q by taking into account the foreign key constraints on the global relations appearing in the atoms. Such an expansion is performed by viewing each foreign key constrains $r_1[X] \subseteq r_2[Y]$, where X and Y are sets of h attributes and Y is a key for r_2, as a logic programming rule

$$r_2'(\vec{X}, f_{h+1}(\vec{X}), ..., f_n(\vec{X})) \leftarrow r_1'(\vec{X}, X_{h+1}, ..., X_m)$$

where each f_i is a Skolem function, $\vec{X}$ is a vector of h variables, and the foreign key are the first h ones. Each r_i' is a predicate, corresponding to the global relation r_i, defined by the above rules for foreign key constraints, together with the rule

$$r_i'(X_1, ..., X_n) \leftarrow r_i(X_1, ..., X_n)$$

Such a logic programming $\mathcal{P}_\mathcal{G}$ can be used to generate the expanded query $exp_\mathcal{G}(q)$ by performing a partial evaluation which computes all certain answers.

3.2 Data Integration Application

There has been a flurry of research in data integration application systems in past years. Several research projects provide the ability of conflict reconciliation based on knowledge. In such a system, ontologies are used to provide concise and declarative specification of semantic information.

The **Carnot** project [CHS91] uses the global ontology *Cyc* to describe the contents of individual information sources in the form of articulation axioms. Queries using terms of *Cyc* are answered by translating them into local queries in terms of information sources. The **InfoSleuth** project [BBB+97] extends the **Carnot** by using multiple domain ontologies and information brokering for information integration. Agent-based infrastructure is **InfoSleuth**'s special feature for data gathering and sharing.

The **SIMS** project [AKS96] provides intelligent access to heterogeneous distributed information sources through a domain model (ontology) which is described in the LOOM Description Logic system. Each element of individual sources has links to some elements in the domain model. A set of operators are defined for query reformulation. The **OBSERVER** project [MKSI96] uses multiple domain ontologies for describing information sources. Brokering across the domain ontologies is enabled by inter-ontology relationships such as synonyms, hyponyms, and hypernyms among terms.

We briefly discuss two salient integration systems: The **TSIMMIS** project [CGMH+94] and the **Information Manifold** project [LSK96], and make a comparison between them. TSIMMIS exports data sources as objects in mediators using Object-Exchange Model (OEM) and Mediator Specification Language (MSL). Queries are effectively processed by the mediators. The following example shows the approach in TSIMMIS.

Example 1. Consider an integrated information system about employees of a company. There are three sources, $v_1(E, P, M)$ producing employee-phone-manager information, $v_2(E, O, D)$ producing employee-office-department information, $v_3(E, P)$ producing employee-phone information for toy department. Using MSL rules, a mediator med that uses these three sources and exports two types of objects:

- Employee-phone-office objects with label *epo*;
- Employee-department-manager objects with label edm; as follows,

1. `<f(E) epo {<name E> <phone P>}>@med :- <emp {<name E> <phone P>}>@source1`
2. `<f(E) epo {<name E> <phone P>}>@med :-<emp {<name E> <phone P>}>@source3`
3. `<f(E) epo {<name E> <office O>}>@med :-<emp {<name E> <office O>}>@source2`
4. `<edm {<name E> <dept D> <mgr M>}>@med :-<emp {<name E> <mgr M>}>@source1 AND <emp {<name E> <dept D>}>@source2`

The following Datalog rules capture much of the content of the MSL rules above:

```
epo(E,P,O) :- v1(E,P,M) , v2(E,O,D).
epo(E,P,O) :- v3(E,P), v2(E,O,D).
edm(E,D,M) :- v1(E,P,M), v2(E,O,D).
```

It is easily to see that query over a mediator can be processed by an unfolding strategy.

□

Information Manifold uses the CLASSIC Description Logic to describe a global schema, i.e., world-view. Sources are defined as views of the global schema. Queries posed against the global

schema are answered by the Bucket algorithm as shown in following example.

Example 2. Suppose the integration system has the same three sources as in Example 1. In this system, the global schema has 5 predicates: *emp(E)* as employee, *phone(E,P)* as phone P of employee E, *office(E,O)* as office O of employee E, *mgr(E,M)* as manager M of employee E, and *dept(E,D)* as department D of employee E. The definition of the three views are:

```
v1(E,P,M) :- emp(E), phone(E,P), mgr(E,M).
v2(E,O,D) :- emp(E), office(E,O), dept(E,D).
v3(E,P)   :- emp(E), phone(E,P), dept(E,toy).
```

The query: `q(P,O) :- phone(sally, P), office(sally, O)` can be answered by producing a bucket containing possible view atoms for each atom in the query body, and then generating a union of contained queries over sources as shown below:

```
answer(P,O) :- v1(sally,P,M), v2(sally,O,D).
answer(P,O) :- v3(sally,P), v2(sally,O,D).
```

□

Both systems use logic-based technology. The fundamental difference is that TSIMMIS uses GAV modelling approach, while Information Manifold uses LAV modelling approach. Another salient architectural difference is that TSIMMIS allows more than one layers of mediators, while Information Manifold only deals with one layer of relationship between the global schema and sources.

4 Schema Manipulation: from Schematical Structure to Semantics

The key to data integration system is to specify the mapping between sources and the global schema. It involves the exploration of the semantics of data residing in sources as well as of the global schema. Schema manipulation including *schema integration and merging*, *schema matching*, *schema mapping*, and *generic model management* represents the efforts in resolving semantic heterogeneity among various types of database schemas. Solutions developed so far range from exploiting schematical structures to explicitly representing semantics in a formal framework.

4.1 Schema Integration and Merging

Problems of schema integration and schema merging emerged from earlier database design context such as view integration from a set of external views. Later, data gathering and sharing in federated database systems call for a global integrated schema in the same spirit. [BLN86] made a comprehensive survey about schema integration in later 80's. Its analysis about the causes of schema diversity and the proposed criteria for a global integrated schema shed light on later efforts in data integration.

In [BLN86], the causes of schema diversity are classified into: (i) different perspectives (Figure 10(a)); (ii) equivalent constructs (Figure 10(b)); (iii) incompatible design specification (Figure 10(c)). For two representations R_1 and R_2 in different schemas, several types of semantic relationships which can exist between them are classified into: (i) identical relationship, meaning R_1 and

R_2 are exactly the same; (ii) equivalent relationship, meaning R_1 and R_2 are exactly the same up to equivalent constructs; (iii) compatible relationship, meaning R_1 and R_2 are neither identical nor equivalent but not contradictory; (iv) incompatible relationship, meaning R_1 and R_2 are contradictory because of the incoherence of the specification. For the equivalent relationship between R_1 and R_2, it can be classified further into: (i) behavioral equivalence, meaning R_1 is equivalent to R_2 for every instance up to every query; (ii) mapping equivalence, meaning there is a one-to-one correspondence between the instances of R_1 and R_2; (iii) transformation equivalence, meaning R_1 can be obtained from R_2 by applying a set of atomic transformations. A global integrated schema may be tested against the following qualitative criteria:

- Completeness and Correctness. The integrated schema must contain all concepts present in component schema correctly.
- Minimality. The same concept must be represented only once.
- Understandability. The integrated schema should be easy to understand for the designer and user.

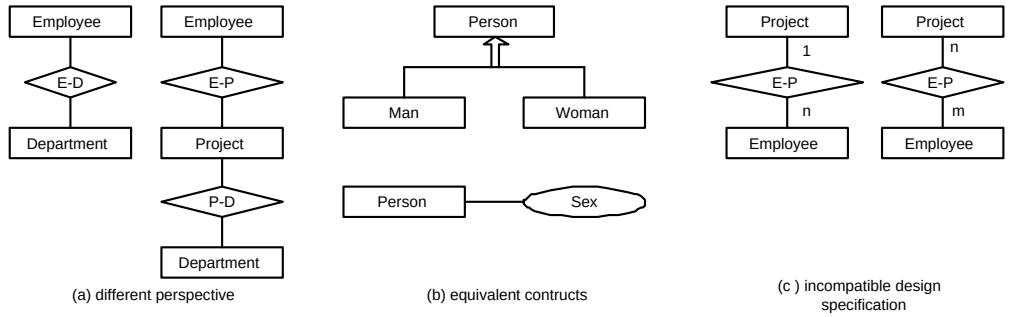

Figure 10: Various causes of schema diversity.

Methods surveyed in [BLN86] are a mixture of techniques involving exploring and resolving conflicts from naming to structures. In contrast, [SP94] uses the *Real World State* as the semantics of the elements and pieces of structures of schemas for integration. After user specifies his knowledge about a set of schemas by correspondence assertions, a tool can integrate the set of schemas into a global schema following integration rules which resolve conflicts. The approach has the flavor of making use of the meaning of data to drive the integration process instead of "guessing" through the names of elements or internal structures in schemas. Here is an example to demonstrate how the principle in [SP94] works.

Example 3. In Figure 10(a), the different perspective causes the different structures of two schemas. In order to integrate them, user can specify the following assertions between elements of two schemas indicating that two elements have same *Real World Semantics* if they are asserted to be equivalent.

Employee≡*Employee, Department*≡*Department,*

Employee-(E-D)≡Employee-(E-P)-Project-(P-D).

Then, following a set of integration rules, one can integrate the two schemas into one which is the same schema exactly as the one shown on the right side in Figure 10(a).

□

There is a class of schema heterogeneity called *schematic discrepancy.* [KLK91] shows the problem as that one database's data (values) correspond to meta-data (schema elements) in others. The following example shows how this situation happens.

Example 4. Consider three stock databases. All contain the closing price for each day of each stock in the stock market. The schemas for the three databases are as follows:

database Company A: relation *Stock-Price(date, stkCode, clsPrice).*
database Company B: relation *Closing-Price(date, stk1, stk2, ..., stkn).*
database Company C: relation *Stock1(date, clsPrice), Stock2(date, clsPrice),..., Stockn(date, clsPrice).*

The schemas above show that the stkCode values in Company A database are the names of the attributes and relations in the other databases, e.g., Company B and Company C up to a name mapping.

□

In order to deal with the schematic discrepancy, [KLK91] defines the universe of databases to be a tuple of relational databases as follows:

u= (db_1: (r_{11}:{(a_{111}:O_{111},...)...},
r_{12}:{(a_{121}:O_{121},...)...}...),
db_2: (r_{21}:{(a_{211}:O_{211},...)...},
r_{22}:{(a_{221}:O_{221},...)...}...)
⋮
)

Each relational database db_i in the above tuple is a tuple of relations. Each relation r_{ij} in each database is a set of tuples, and each tuple in a relation is a tuple of objects. These objects in a tuple are atomic. Given a query language over objects, e.g., atomic values, sets, and tuples, which allows variables appear as names of attributes or relations, one can express a query like "Did any stock ever closed above 200?" in the context of Company B and Company C as follows:

?.CompanyB.Closing-Price(.X>200).
?.CompanyC.X(.clsPrice>200).

where X is a variable. The query language contains high order features, and a global unified view can be defined in terms of sources using the high order query language.

4.2 Schema Matching

Compared to schema integration, *schema matching* does not produce a unified schema from a set of component schemas, but identifies semantical relationships between schemas. In *schema matching*, a matcher takes two schemas as input and produces a mapping between elements of the two schemas that correspond "semantically" to each other. A taxonomy which covers a variety of matching techniques is proposed in [RB01]. In particular, matchers are classified into schema- and instance-level, element- and structure-level, language- and constraint-level, and individual and combination methods. It is worth noting that all methods surveyed are semi-automatic which need the intervention of users.

Specifically, Cupid [MBR01] discovers mappings between schema elements based on their names, data types, constraints, and schema structures. Using schema tree and schema graph as the unifying internal representation of a spectrum of schemas, Cupid identifies similarity between elements in two steps: linguistic-based matching follows structure-based matching. What Cupid assumes about "semantical correspondence" is that the names of the elements convey linguistic similarity and the structures of the representation help to propagate similarity along graph edges. Matching is computed, essentially, by "guessing".

There are others schema matching efforts in the same spirit as Cupid does: exploring the structures of schemas without paying attention to the real world semantics of the schemas. [MGMR02] presents a graph matching algorithm called Similarity Flooding and explores its usability for schema matching. The approach converts schemas into directed labeled graphs and uses fixpoint computation to determine the matches between corresponding nodes of the graphs. For ontologies with rich semantics, [NM01] proposes the Anchor-PROMPT algorithm which takes as input a set of anchors – pairs of related terms in the two ontologies, producing pairs of other related terms. Anchor-PROMPT assumes the structure conveys similarity. The COMA [DR02] does not invent any new matching algorithm. Instead it develops a framework to combine multiple matchers in a flexible way. Multiple matching techniques can be plugged in COMA system to produce a composite matching results for input schemas. Hence the "semantic" interpretation of schema elements depends on underlying matchers.

4.3 Schema Mapping

Although *schema matching* intends to discover semantical relationships between elements of schema, most of the current techniques merely generate a set of syntactic correspondences each of which is between a single element of one schema and a single element of another schema. It is fallen far short of generating executable mappings in terms of query languages. Moving one step further, *schema mapping* [MHH00] discovers the formal queries over input schemas in accordance with the value correspondences specified by users or matchers. A query or set of queries maps a source database into a different, but fixed, target database. [MHH00] proposes the schema mapping problem arguing that both appropriate outside information about schemas such as value correspondences and inside information such as integrity constraints are indispensable for query discovery. For the setting in which both source and target schema are of relational schema, [MHH00] presents a query discovery algorithm which sifts through a set of alternative join paths in the source schema to find meaningful mappings which cover a set of pre-specified value correspondences.

Considering *nested relational structures* such as XML databases, [PVM$^+$02] extends the query discovery work to cover the cases of *nested integrity constraints* in target schema. Traditional *chase*

technique which produces all logical relations among elements within individual schemas is the core component of the query discovery process. The mapping algorithm in [PVM$^+$02] looks at all pairs of source and target logical relations to find low level executable mappings (i.e. formal queries) which interpret the high level general mappings (i.e. value correspondences). The following example is a schema mapping scenario appearing in [PVM$^+$02].

Example 5. Given a source schema and a target schema and their integrity constraints in Figure 11. The value correspondence v1, v2, v3 indicate the user's intention about the correspondences between elements in the source and target schemas. Low level executable mappings which interpret the value correspondences are generated by the tool, Clio. The following expression shows one of them:

$\forall$*(g*$\in$*expenseDB.grants)(c*$\in$*expenseDB.companies)(p*$\in$*expenseDB.projects)*
[g.grant.cid=c.company.cid$\wedge$*g.grant.proj=p.project.name*$\Rightarrow$
$\exists$*(c'*$\in$*statDB)(o*$\in$*c'.cityStat.orgs)(f*$\in$*o.org.fundings)(f'*$\in$*c'.cityStat.financials)*
f.fund.pi=g.grant.pi$\wedge$*o.org.name=c.company.cname*$\wedge$*f'.financial.aid=f.fund.aid]*

□

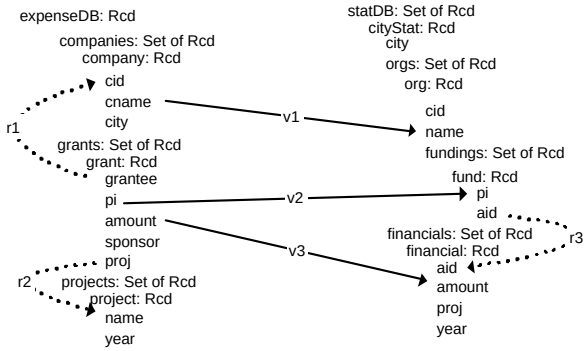

Figure 11: A schema mapping scenario

4.4 Generic Model Management

Realizing that manipulating formal description, or models, is a key requirement to many information systems, [Ber03] proposes a new meta data management paradigm called *generic model management* which treats models and mappings as abstractions that can be manipulated by model-as-a-time and mapping-as-a-time operators. A *model* is defined to be a set of objects, each of which has properties, has-a relationships, and associations. A *model* is assumed to be identified by its root object and includes exactly the set of objects reachable from the root by paths of has-a relationships. Given two models M_1 and M_2, a *morphism* over M_1 and M_2 is a binary relation over the objects of the two models. That is, it is a set of pairs <o_1, o_2> where o_1 and o_2 are in M_1 and M_2 respectively. A *mapping* between models M_1 and M_2 is a model, map_{12}, and two morphisms, one between map_{12} and M_1 and another between map_{12} and M_2. An example of model and mapping is shown in Figure 12(a).

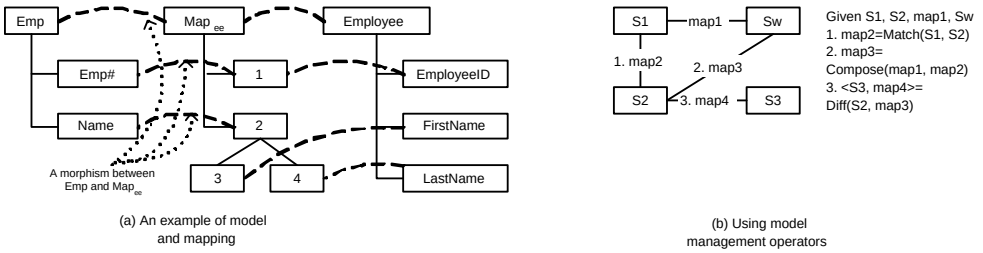

Figure 12: Model, mapping and generic operators.

By defining a set of generic model management operators including *Match*, *Compose*, *Diff*, *ModelGen*, and *Merge*, one can use them in a specific application as shown in the following example [Ber03].

Example 6. Suppose we are given a mapping map1 from a data source S1 to a data warehouse Sw, and want to map a second source S2 to Sw, where S2 is similar to S1. See Figure 12(b). First we call Match(S1, S2) to obtain a mapping map2 between S1 and S2, which shows where S2 is the same as S1. Second, we call Compose(map1, map2) to obtain a mapping map3 between S2 and Sw, which maps to Sw those objects of S2 that correspond to objects of S1. To map the other objects of S2 to Sw, we call Diff(S2, map3) to find the sub-model S3 of S2 that is not mapped by map3 to Sw. We can then call other operators to generate a warehouse schema for S3 and merge it into Sw.

□

Although an implementation of these abstractions and operators, called a *model management system*, could offer an order-of-magnitude improvement in programmer productivity for meta data applications, it is by no means an easy task to generate mappings between models automatically without explicit semantic representation.

4.5 Managing Meta-data and Context

Turning to the meta-level information of schemas, [KS96a, KS96b] explore approaches based on the capture and representation of meta-data, contexts, and ontologies to manage the semantic heterogeneity in Global Information Systems. Given two objects O_1 and O_2 residing in different models, the semantic proximity between them is defined by the 4-tuple:

$$semPro(O_1, O_2) =< Context, Abstraction, (D_1, D_2), (S_1, S_2) > .$$

Where D_i is the domain of O_i and S_i is the state of O_i. An explicit though partial context representation, which is a collection of coordinates using terms from domain-specific ontologies, is introduced as follows:

$$Context =< (C_1, V_1)(C_2, V_2), ..., (C_k, V_k) >.$$

Where C_i is a contextual coordinate denoting an aspect of context from a domain ontology; V_i is the value of a coordinate, and is represented as a variable or a set with or without its associated context. We shall informally explain the meaning of the context expression by using the following example [KS96b].

Example 7. Suppose we want to represent the information relating publications to employees in a database. Let PUBLICATION and EMPLOYEE be objects in a database. The definition context of HAS-PUBLICATION can be defined as:

C_{def}*(HAS-PUBLICATION)=<(article,PUBLICATION) (author,EMPLOYEE* $\circ$ *<(affiliation, {research})>)>.*

Where $\circ$ denotes association of a context with an object EMPLOYEE. Note that, the same thing can be expressed in a Description Logic as follows:

C_{def}(HAS-PUBLICATION)=(AND HAS-PUBLICATION (ALL article PUBLICATION)
(ALL author (AND EMPLOYEE (ALL affiliation (ONE-OF research))))).

The terms *article, author, affiliation,* and *research* are from a domain ontology.
Likewise, if we want all the articles whose titles contain the substring "abortion" in them. This can be expressed in the following query context:

C_q=<(article, X $\circ$ <(title, {y |substring(y)="abortion"})>)>.

□

The context expressions enable reasoning about them. Also, they enable an information source to export global objects with context description to a data integration system. A schema correspondence is defined as follows to express the associations between an information source and its exported global objects:

$$schCor(O_G, O) =< O_G, \{C_i \mid C_i \in C_{def}(O)\}, O, attr(O), M >.$$

Where O_G is the exported global object of an object O in the database. The attributes of the object O_G are the contextual coordinates of the definition context $C_{def}(O)$. The mapping M stores the association between the contextual coordinate C_i and the attribute A_i of object O whenever there exists one.

The semantic similarity between objects in different schemas is classified into a taxonomy including *semantic resemblance, semantic relevance, semantic relationship, semantic equivalence, and semantic incompatibility* based on the context, the domains of the objects, and the states of the objects. The association of objects with context enables the capture of the semantic content of the information present in various databases so that the reconciliation of conflicts can be achieved at an intensional level.

The effort of overcoming semantic heterogeneity by considering context representation gives insights to the explicit representation of semantics in terms of correspondence. However, the approach lacks the analysis about the computational complexity of reasoning. Also, its representation is merely a variation of Description Logics, but it lacks simplicity and elegance as opposed to the Description Logics.

5 The Semantic Web towards Machine-Understandability

Data on the Web raise more challenges to interoperation, exchange, and integration than data in traditional structural databases. Today we are limited in our ability to effectively use information on the Web despite the ubiquity of tightly interconnected data and processes. Current Web was designed primarily for human interpretation and use. Nevertheless, interoperable applications exist in B2B and e-commerce areas by primarily exploiting hand-coded APIs to extract and locate information from HTML syntax. Data on the current Web are not understandable by computers. The semantic web [BLHL01] is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications [MSZ01].

Currently, people are developing new markup languages inspired by technology from AI, such as the proposals of DAML+OIL and its latest version OWL [DS03] web ontology language. These languages have a well-defined semantics and enable the markup and manipulation of web contents. A fundamental component of the semantic web will be the markup of web data exploiting web ontologies to facilitate sharing, reuse, composition, and mapping for various applications.

The OWL web ontology language is intended to define and instantiate web ontologies. An OWL web ontology may contain descriptions of classes, properties, and their instances. Given such a web ontology, the OWL formal semantics specifies how to derive its consequences, i.e., facts not explicitly presented in the ontology but entailed by its semantics. An ontology differs from an XML schema or DTD in that it is a knowledge representation, not a message format. Most XML-like web standards consist of a combination of message formats and protocol specifications. The formats have been given an operational semantics. In contrast, an OWL ontology allows for reasoning outside a operational environment. The following example demonstrates a use of OWL and the machine-understandable content.

Example 8. A typical OWL ontology begins with a namespace declaration which is the precise indication of what specific vocabularies are being used. The declaration is similar to the following:

```
<rdf:RDF
    xmlns="http://www.cs.toronto.edu/Academic-Organization#"
    xmlns:owl="http://www.w3c.org/2002/07/owl#"
    xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3c.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3c.org/2000/10/XMLSchema#"
>
```

Classes and properties are described using the primitive constructs of OWL language. Each of the constructs has a formal semantics supplied by the underlying logical formalism. A tiny portion of an academic organization ontology is shown as below:

```
<owl:Class rdf:ID="Student">
    <rdfs:subClassOf  rdf:Resource="&Person"/>
    <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty  rdf:Resource="&enrollIn"/>
          <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
        </owl:Restriction>
     </rdfs:subClassOf>
```

```
</owl:class>
```

Using the web ontology, one can publish annotated web data readily for machines to consume, as follows.

```
<rdf:RDF
   xmlns:Ac-Onto="http://www.cs.toronto.edu/Academic-Organization#"
   xmlns="http://www.cs.toronto.edu/~yuana/yuana.owl#"
>
<Ac-Onto:Student  rdf:about="#yuana">
   <Ac-Onto:FirstName>Yuan</Ac-Onto:FirstName>
   <Ac-Onto:LastName>An</Ac-Onto:LastName>
   <Ac-Onto:majorIn>Computer Science</Ac-Onto:majorIn>
</Ac-Onto:Student>
```

□

The two-level mapping from instance data to commonly agreed concepts of ontologies eliminates the semantic heterogeneity of independent data sources. Integration [HBLM02] is readily performed as long as standardization is widely accepted. Other large volumes of data, however, do not fit in this picture because the existing "legacy systems," such as relational databases and a variety of semi-structured and unstructured data. Therefore a framework for bridging the gap between legacy systems and the semantic web is called for to make the whole vision realizable.

6 From Knowledge Representation to Formal Ontology

The markup of web data using AI-inspired language reminds us that achieving the ambitious goal of building machine-understandable information will rely on knowledge manipulation techniques. Turning to knowledge representation, we will look at languages related to information systems as well as principles of building ontologies.

6.1 Representing Knowledge about Information Systems

An effort for representing knowledge about information systems results in the Telos [MBJK90] language and its implementation, ConceptBase [JGJS95], a deductive object base for meta data management based on a dialect of Telos, O-Telos. Telos represents knowledge about a variety of worlds related to an information system, such as the subject world (application domain), the usage world (user models, environments), the system world (software requirements, design), and the development world (teams, methodologies).

The most distinct feature of Telos is the capability of metamodelling. Everything in a Telos information base is a proposition, and there are two types of propositions: *individuals* and *attributes*. Propositions are organized along three dimensions, referred to as the *aggregation*, *classification*, and *generalization* dimensions. The classification dimension calls for each proposition to be an instance of one or more generic propositions or classes. Therefore, metaclasses can be always defined to serve as the more abstract classes than one-level below. There are ω-classes with instances along all layers. Figure 13 appearing in [JGJS95] shows the structure of the classification dimension as well as the generalization dimension, with sample proposition at various levels.

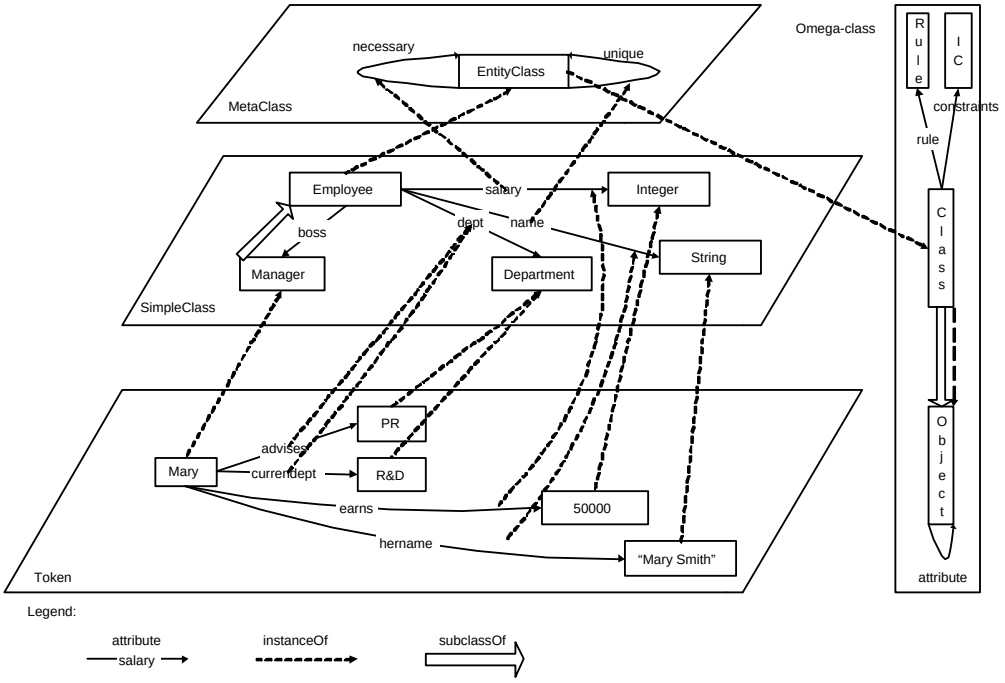

Figure 13: Graphical view of Telos knowledge base of an example.

In addition to propositions, Telos allows specifying *integrity constraints* and *deductive rules* to associate with propositions. The following example shows how to define propositions and their associated constraints and rules in Telos language.

Example 9. The following text defines some propositions as well as constraints and rules in Telos language.

```
TELL SimpleClass Employee IN EntityClass WITH
    attribute
          dept:  Department;
          boss:  Manager;
    necessary
          salary:  Integer
    unique
          name:  String
    integrityConstraint
          salaryIC: $ forall m/Manager x, y/Integer
                  (this boss m) and (this salary x) and (m salary y) ==>
                  (x <= y) $
    deductiveRule
          bossrule:  $ forall t/Manager
                  (exists d/Department (this dept d) and (d head t)
                  or exists m/Manager (this boss m) and (m boss t) ) ==>
                  (this boss t) $
END
```

□

With the development of knowledge representation language, a family of Description Logics emerged and found their way into various tasks of representing knowledge.

6.2 Description Logics Theory and Application

[BN03] introduces a family of Description Logics called $\mathcal{AL}$-language. Generally speaking, Description Logics is the formalism that represent the knowledge of an application domain (the "world") by first defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects and individuals occurring in the domain (the world description). Elementary descriptions are atomic concepts and atomic roles. Complex descriptions can be built from them inductively with concept constructors. Different Description Languages are distinguished by the constructors they provide. Concept descriptions in $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}][\mathcal{I}]$ are formed according to the syntax rules in the Table 3.

Syntax Rules	**Description of Rules**	**Semantics**
A	*(atomic concept)*	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\top$	*(universal concept)*	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
$\bot$	*(bottom concept)*	$\bot^{\mathcal{I}} = \emptyset$
$\neg A$	*(atomic negation)*	$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
$C \sqcap D$	*(intersection)*	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\forall\, R.C$	*(value restriction)*	$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
$\exists R.\top$	*(limited existential quantification)*	$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$
$C \sqcup D$	*(union)* $[\mathcal{U}]$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists\, R.C$	*(full existential quantification)* $[\mathcal{E}]$	$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
$\geq n\, R$	*(number restriction)* $[\mathcal{N}]$	$(\geq n\, R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\rvert \geq n\}$
$\leq n\, R$	*(number restriction)* $[\mathcal{N}]$	$(\leq n\, R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\rvert \leq n\}$
$\neg\, C$	*(negation of arbitrary concept)* $[\mathcal{C}]$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
R^-	*(inverse role)* $[\mathcal{I}]$	$(R^-)^{\mathcal{I}} = \{(b,a) \mid (a,b) \in R^{\mathcal{I}}\}$

Table 3: Syntax and semantics of $\mathcal{ALUENCI}$.

In order to define a formal semantics of the above Description Language, we consider an interpretation $\mathcal{I}$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concept descriptions by the inductive definitions as shown in the last column of Table 3.

Description Logics has been incorporated into various aspects of data management systems. [CLN99] describes an effort to unify various class-based data representation formalisms using the Description Logics – $\mathcal{ALUNI}$. The relationships between class-based data representation formalisms and $\mathcal{ALUNI}$ are established by first formalizing the representation, second defining a translation function from the formalism obtained in the first step to $\mathcal{ALUNI}$, and then proving that the translation function preserves information. One fundamental reason for regarding $\mathcal{ALUNI}$ as a unifying framework for class-based representation formalisms is that reasoning in $\mathcal{ALUNI}$ is hard, but decidable. Three class-based formalisms, Frame-Based systems, Entity-Relationship model, and object-oriented model are studied in [CLN99]

[BLR03, Bor95] investigate the applications of DLs in data management and databases. They show that DLs are useful not only for specifying database schemas, but also for asserting incomplete information, for obtaining intensional answers, for stating rules and constraints, and other applications in data management.

DLs is also the underlying formalism of the semantic web. The idea behind applying DLs to the semantic web is related to the need of representing and reasoning on ontologies. One of the basic problems in the development of techniques for the semantic web is the interoperation of ontologies. [CL93] proposes a logical approach to the problem of both expressing interschema knowledge and reasoning about it with a flavor of DLs in a early stage. However, [CGL01] argues that direct use of DLs is inadequate for capturing the mapping between different ontologies, and more flexible mechanisms are necessary. One of the mechanisms is based on the notion of *query* which specifies the relationships between concepts and views in different ontologies. Most recently, [BS03] extends the DLs to *Distributed DLs* to handle complex mappings between domains of different data sources, through the use of "*bridge rules*". One of the interesting results obtained is the translation scheme of DDL reasoning to ordinary DL reasoning, which is applicable to a large class of description logics.

As a knowledge representation language, a specific DL is capable to represent the semantics of an application domain. For data integration from heterogeneous sources, a specific DL is useful to achieve the integration at the conceptual level [CG$^+$98]. The DLs can be used to define the domain ontology of each application, and then schemas can gain real world semantics by corresponding to the domain ontology. The section to follow will discuss a principled methodology for building ontologies.

6.3 Properties of Formal Ontology

In Philosophy, ontology is the study of what is. It studies various kinds and structures of objects, properties, events, processes, and relations in every area of reality [SW01]. It has sought the definitive and exhaustive classification of entities in all aspects of being. The term "ontology" has been borrowed by computer scientists with the development of artificial intelligent and other sub-fields such as software engineering and database management.

From the very start, ontology has been recognized as collections of terms with associated axioms designed to constrain unintended interpretations and to enable the derivation of new information from ground facts. We have seen much progress in the study of automated reasoning mechanisms over ontologies and knowledge bases; however, there is a high degree of arbitrariness for modelling subject matters in terms of symbolic structures. For example, it has been seen that, after database management technology had begun to stabilize, the far more important and subtle problem of conceptual modelling still remained. The various ad hoc and inconsistent modelling techniques have led to the many practical problems of database integration we face today. This situation raises at least two questions challenging us: (i) how can we conduct a consistent conceptual analysis to build, once for all, a common, robust, and reusable ontology? (ii) how can we make legacy systems with different conceptual models but overlapping semantics work together by referring to the common world to which they all relate? While the second question is the main theme of the whole research, we will discuss methodology related to the first question in this section.

[Gru93] is often credited for first defining ontology as a specification of a conceptualization. Although it leaves too many possible interpretations, a collection of taxonomies with a set of axioms remain the primary one. In order to conduct a consistent conceptual analysis to build a good tax-

onomy, one needs a general, domain-independent methodology that provides guidance not only on what kinds of ontological decisions need to be made, but on how these decisions can be evaluated.

OntoClean [GW02, WG01, GWP00] is this kind of methodology to validate taxonomies by exposing inappropriate and inconsistent modelling choices. The methodology is based on four fundamental ontological notions: *identity, unity, rigidity,* and *dependency*. The behavior of a property is represented with respect to these notions by means of a set of meta-properties which impose some constraints on the way subsumption is used to model a domain.

Formally, let M denote a primitive meta-properties, let $\phi^{+M}, \phi^{-M}, \phi^{\sim M}$ denote the property ϕ *carrying* the meta-property M, *not carrying* M, and *anti* the meta-property M, respectively. In a modal logic, where $\Box\phi$ means ϕ is necessary true, $\Diamond\phi$ means ϕ is possible true, a *rigid* $(+R)$ property is a property that is essential to all its instances, i.e. a property ϕ such that: $\Box(\forall x t \phi(x,t) \rightarrow \Box t' \phi(x,t'))$. Similarly, a *non-rigid* $(-R)$ property is a property that is not essential to some of its instance, and an *anti-rigid* $(\sim R)$ property is a property that is not essential to all its instances. They can be described as the formulas $\Diamond(\exists x \phi(x,t) \wedge \Diamond \exists t' \neg\phi(x,t'))$ and $\Box(\forall x t \phi(x,t) \rightarrow \Diamond \exists t' \neg\phi(x,t'))$, respectively.

An *identity condition (IC)* is a sameness formula Σ that satisfies

$$\Box(E(x,t) \wedge \phi(x,t) \wedge E(y,t') \wedge \phi(y,t') \wedge x = y \rightarrow \Sigma(x,y,t,t')), \text{ or}$$
$$\Box(E(x,t) \wedge \phi(x,t) \wedge E(y,t') \wedge \phi(y,t') \wedge \Sigma(x,y,t,t') \rightarrow t = y)).$$

Any property carries an IC $(+I)$ iff it is subsumed by a property supplying that IC. A property ϕ supplies an IC $(+O)$ iff (I) it is rigid; (II) there is an IC for it; and (III) the same IC is not carried by all the properties subsuming ϕ. Any property carrying an IC is called a sortal.

As for the notion of unity, an object x is a whole under ω iff ω is relation such that all the members of a certain division of x are linked by ω, and nothing else is linked by ω. A property ϕ carries a unity condition $(+U)$ iff there exists a single relation ω such that each instance of ϕ is necessarily a whole under ω. A property has anti-unity $(\sim U)$ if every instance of the property is not necessarily a whole.

Let $P(y,x)$ denote the part relation, and $C(x,y)$ denote the constitution relation, a property ϕ is externally dependent $(+D)$ on a property ψ if, for all its instances x , necessarily some instance of ψ must exist, which is not a part nor a constituent of x:

$$\forall x \Box(\phi(x) \rightarrow \exists y \psi(y) \wedge \neg P(y,x) \wedge \neg C(y,x).$$

If follows from these definitions that if ϕ and ψ are properties, then the following constraints hold:

$\phi^{\sim R}$ must subsume $\psi^{\sim R}$
ϕ^{+I} must subsume ψ^{+I}
ϕ^{+U} must subsume ψ^{+U}
$\phi^{\sim U}$ must subsume $\psi^{\sim U}$
ϕ^{+D} must subsume ψ^{+D}
Properties with incompatible ICs/UCs are disjoint

The various combinations of meta-properties result in a formal ontology of properties shown in Figure 14(a). The basic property kinds are shown in Table 4. One of the principal roles of the

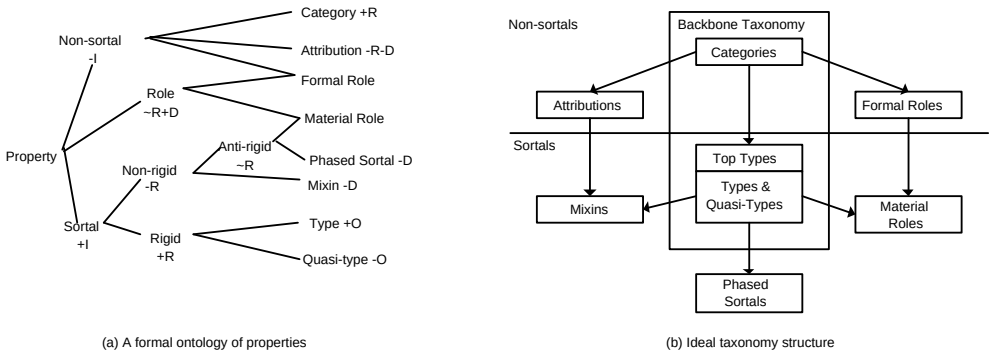

Figure 14: Ontology-based modelling principles.

O	I	R	D	
+	+	+	+-	Type
-	+	+	+-	Quasi-Type
-	+	-	-	Mixin
-	+	~	+	Material Roles
-	+	~	-	Phased Sortal
-	-	+	+-	Category
-	-	~	+	Formal Roles
-	-	-	-	Attribution

Table 4: Basic property kinds.

formal ontology of properties is to impart structure on an ontology. In general, the ideal structure of a clean taxonomy based on the property kinds is shown in Figure 14(b), where *backbone properties* constitute the *backbone taxonomy*.

An implementation of the methodology results in a question/answer knowledge-based system which can help modelers to make modeling assumptions clear and to produce well-founded taxonomies by verifying the consistency of a taxonomy based on the constraints among meta-properties. The following short example appearing in literature demonstrates an application of the methodology.

Example 10. There are two well-known ontologies, WordNet and Pangloss. Both define the entities, *Physical object* and *Amount of matter*. But they have different subsumption relationships:

- **WordNet**: *Physical object* isA *Amount of matter*.
- **Pangloss**: *Amount of matter* isA *Physical object*.

Intuitively, we can assign meta-properties to *Physical object* and *Amount of matter* as follows:

- *Physical object*: **+O+U+R-D**
- *Amount of matter*: **+O~U+R-D**

By this analysis, we may conclude that both *Physical object* and *Amount of matter* are type, and both should be at the top-level.

□

6.4 Standard Upper Ontology

A great deal of effort has been put into designing standard upper ontologies to explicitly represent entire human knowledge as complete as possible. We refer to Cyc [Cor03], and SUO [IEE03] as examples. A distinct feature of Cyc is that microtheories are used to resolve conflicts. A microtheory or a context is a set of assertions that have a shared set of assumptions on which the truth of assertions depends. The assertions within a microtheory must be mutually consistent, while assertions in different microtheories may be inconsistent. The application of microtheory demonstrates that the context of elements in different models plays a pivotal role for the identification of semantic similarity. The WordNet [MFT+03] is a lexical database for English language. It becomes useful in the development of ontologies for picking up right terms.

7 Research Direction and Challenges

From data integration to the semantic web, we have acquired a great deal of knowledge as well as various techniques about dealing with heterogeneity. Each solution that has been seen so far overcomes semantic heterogeneity from a specific perspective. Incorporating features of knowledge representation and database query processing, we believe that a systematic study and representation of data semantics in terms of correspondence will benefit data integration as well as the construction of the semantic web. What to follow is the preliminary description about the direction and challenges to approach the problem of data semantics in terms of correspondence.

7.1 Establishing and Maintaining Semantic Correspondence

Database practices today show that the meanings of data amount to a set of composite correspondences. Consider the simple database design example in Section 2. An Entity-Relationship schema for a university database is a model of some parts of the university world. A relational schema of the same database constitutes a model of the Entity-Relationship schema. A website that makes student information available is a model of the relational schema. In other words, data semantics amounts to a continuum of correspondences, anchored in the application. Correspondence continuum as shown in Figure 15 is investigated in [Smi87]. It shows that a set of genuine semantic relations including specification, encoding, internalisation, externalisation, implementation, and representation constitute a correspondence continuum starting from linguistic structures, midway across models with richer semantics, at some point, reaching the states of affairs in the world that the original structures were genuinely about. We will adopt the correspondence continuum framework, focusing on various database schemas. We treat the real world states as a formal domain ontology.

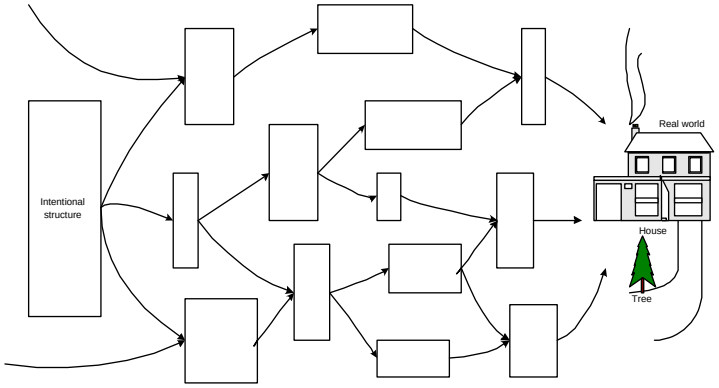

Figure 15: Correspondence continuum.

One immediate benefit from the maintenance of semantic correspondences is to advance the schema mapping discovery to a conceptual level. Apparently, establishing and maintaining semantic correspondence bring about many challenges in the development of language and reasoning mechanisms.

7.2 Representing and Reasoning about Correspondences

There is a variety of mapping specification languages ranging from simple value correspondences to formal logical expressions. For automation, a formal mapping specification language is needed as well as the reasoning ability about mappings. [CL93] proposes a logic approach to the problem of both expressing inter-schema knowledge and reasoning about it. In particular, schemas are described in a class representational language and inter-schema knowledge is described in logic assertions between class expressions from different schemas. Reasoning tasks such as inter-schema consistency and integrated query answering can be performed making use of the knowledge network built from the class representation of the schemas and the assertions. Specifically, reasoning is reduced to a logical implication problem.

[C+01] describes a data integration system using a conceptual view and a set of reconciliation correspondences to load data from operational databases into a data warehouse. Both source schemas and data warehouse schema are declaratively specified as views over the conceptual view. The declarative view definitions and reconciliation correspondences serve as mappings providing the vehicle for the generation of a mediator.

[MB+02] offers a framework for defining representation of mappings with associated semantics. The analysis of several classes of applications which heavily reply on mappings shows that a clear semantics which provides a basis for reasoning about mappings, the ability to accommodate incompleteness, and the ability to allow heterogeneity of models are the desiderata for representation of mappings. Given two models described in some logical languages, the mappings between them are defined as a set of formulas each of which associates an expression of the first model with an expression of the second model by a meaningful operator. In model-theoretic sense, the semantics of a mapping formula is defined by the logical satisfaction of the union of the interpretations of the models (i.e. schemas).

Mapping composition is a key reasoning requirement for maintaining correspondence. [MH03] treats the mapping composition problem in terms of the global-local-as-view (GLAV) specification. it shows that even composing two very simple mappings, the full composition may be an infinite set. Hence complicated algorithm is called for in order to accomplish the mapping composition task. We believe, however, that mapping composition can be fulfilled within a simple framework yet without losing its ability to address a variety of data manipulation problems. Mapping discovery based on the framework of maintaining correspondence continuum amounts to the mapping composition problem in terms of GLAV specification.

Most recently, the peer-to-peer data management system (PDMS) has gained an increasing attention in database community. A PDMS heavily relies on semantic mappings between peers' schemas, and makes use of massive techniques of query reformulation developed in data integration settings, see [HIMT03] and [HIST03]. One of the enduring problems left out by the PDMS is automatic specification of semantic mapping between peers. The explicit presentation of data semantics could be a possible approach to such a problem. In the semantic web research community, people mostly focus on semantic coordination between ontologies which are mainly represented in taxonomies. [B+03] extends the current OWL semantics into contexualizing ontology semantics, arguing autonomy is intrinsic in the semantic web environment, so mappings are indispensable for interoperation. [BSZ03] shifts the semantic coordination problem from computing linguistic and structural similarities to deducing relations between sets of logical formulas representing the meanings of concepts belonging to different ontologies. [FS03] proposes an ontology mapping method based on information flow theory. All of the above applications and approaches involving mappings between different modeling structures compel us to proceed the research line on properties of mapping which becomes the significant interest in capturing data semantics.

8 Concluding Remarks

By studying existing approaches to dealing with autonomy, heterogeneity, and openness, we believe that an explicit representation of data semantics in terms of correspondence and formal ontology could have its own merits. Although we have discussed a methodology for good ontology design, the development and construction of various domain ontologies is still a big open problem, and we conceive that it is beyond the scope of this research. We will consider, however, the dynamic aspects

of domain ontologies in the sense that new concepts and relationships will be added whenever needed because of the insurmountable incompleteness of those ontologies.

In summary, we can see the following as specific deliverables of the thesis by overcoming some of the challenges:

- A framework based on principles of correspondence continuum and semantic encapsulation for addressing the problem of data heterogeneity arising from the multiplicity of data sources.
- A formal correspondence language which enables database designers explicitly express the semantic relations between models hidden inside their minds.
- A formal approach to the problem of circumstantial parameter effect which causes multiple interpretations of a common concept.
- An analysis of algorithmic issues for reasoning about correspondences - in particular, an algorithm of correspondence composition, an algorithm of finding semantic mapping between two schemas with overlap, as well as related properties.

Each of this contributions will constitute the desiderata of my Ph.D. thesis in the field of computer science, specifically, dealing with the problem of data semantics.

References

[AD98] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS'98, Seattle, WA*, 1998.

[AKS96] Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2-3):99–130, 1996.

[B+03] Paolo Bouquet et al. C-OWL: Contextualizing Ontologies. In *ISWC'03*, 2003.

[BBB+97] R. J. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helai, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *SIGMOD'97*, pages 195–206, 1997.

[Ber03] P. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of Methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–264, 1986.

[BLR03] Alex Borgida, Maurizio Lenzerini, and Riccardo Rosati. Description Logics for Data Bases. In *The Description Logic Handbook, Cambridge University Press*, pages 462–485, 2003.

[BN03] Franz Baader and Werner Nutt. Basic Description Logic. In *The Description Logic Handbook, Cambridge University Press*, pages 43–96, 2003.

[Bor95] Alexander Borgida. Description Logics in Data Management . *J. of Knowledge and Data Engineering*, 7(5):671–682, 1995.

[BS03] Alex Borgida and Luciano Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *J. on Data Semantics*, 1:153–184, 2003.

[BSZ03] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: a new approach and an application. In *ISWC'03*, 2003.

[C+01] Diego Calvanese et al. Data integration in data warehouse. *Cooperative Information Systems*, 10(3):237–271, 2001.

[CCGL02] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of CAiSE'02, Toronto, Canada*, 2002.

[CG+98] D. Calvanese, G. D. Giacomo, et al. Information integration: conceptual modeling and reasoning support. In *6th Int. Conf. on Cooperative Information Systems*, pages 280–291, 1998.

[CGL01] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Ontology of integration and integration of ontologies. In *Description Logics*, 2001.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TIMMIS project: integration of heterogeneous information sources. In *Proceedings of 10th IPSJ conference*, 1994.

[CHS91] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24:55–62, Dec 1991.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. In *Proceedings of the International Conference on Intelligent and Cooperative Information Systems, IEEE Computer Society Press*, 1993.

[CLN98] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263, 1998.

[CLN99] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999.

[CM77] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[Cor03] OpenCyc Corp. *OpenCyc.* , http://www.opencyc.org, 2003.

[CV92] S. Chaudhuri and M. Y. Vardi. On the equivalence of recursive and nonrecursive Datalog programs. In *Proc. of PODS'92*, pages 55–66, 1992.

[DR02] H. H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *28th VLDB*, 2002.

[DS03] Mike Dean and Guus Schreiber. *OWL web ontology language reference*. W3C Working Draft 31, http://www.w3c.org/TR/owl-ref, March 2003.

[FS03] Yannis Falfoglou and Marco Schorlemmer. IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory. *J. on Data Semantics*, 1:98–127, 2003.

[Gru93] Thomas R. Gruber. Toward priciple for the design of ontologies used for knowledge sharing. Technical Report KSL 93-04, Stanford University, 1993.

[GW02] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with OntoClean. *Communication of the ACM*, 45(2):61–65, February 2002.

[GWP00] Nicola Guarino, Christopher A. Welty, and Christopher Partridge. Towards ontology-based harmonization of web content standards. In *ER Workshops*, pages 1–6, 2000.

[Hal00] Alon Y. Halevy. Theory of Answering Queries Using Views. In *SIGMOD Record 2000*, 2000.

[Hal01] Alon Y. Halevy. Answering queries using Views: a survey. *VLDB*, 10(4):270–294, 2001.

[HBLM02] James Hendler, Tim Berners-Lee, and Eric Miller. Integrating applications on the Semantic Web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10):676–680, Oct. 2002.

[HIMT03] Alon Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: data managment infracstructure for semantic web application. In *WWW'03*, 2003.

[HIST03] Alon Y. Halevy, Z. G. Ives, D. Suciu, and Igor Tatarinov. Schema Mediation in Peer Data Management Systems. In *ICDE'03*, 2003.

[IEE03] IEEE. *Standard Upper Ontology*. , http://suo.ieee.org, 2003.

[JGJS95] Matthias Jarke, Rainer Gallersdorfer, Manfred A. Jeusfeld, and Martin Staudt. ConceptBase: a deductive object base for meta data management. *J. of Intellegent Information Systems*, 4(2):167–192, March 1995.

[KLK91] Ravi Krishnamurthy, Witold Litwin, and William Kent. Languages features for interoperability of databases with schematic discrepancies. In *ACM SIGMOD*, pages 40–49, 1991.

[Klu88] A. C. Klug. On conjunctive queries containing inequalities. *J. of the ACM*, 35(1):146–160, 1988.

[KS96a] V. Kashyap and A. Sheth. Semantic heterogeneity: role of metadata, context and ontologies. In *M. Papazoglou and G. Schlageter (ed.), Cooperative Information Systems: Current Trends and Directions*, 1996.

[KS96b] Vipul Kashyap and Amit Sheth. Semantic and schematic similarities between database objects:a context-based approach. *VLDB*, 5:276–304, 1996.

[Len02] M. Lenzerini. Data integration: a theoretical perspective. In *PODS*, pages 233–246, 2002.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *PODS'95, San Jose, CA*, 1995.

[LSK96] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, Dec. 1996.

[MB+02] J. Madhavan, P. A. Bernstein, et al. Representing and reasoning about mappings between domain models. In *AAAI*, 2002.

[MBJK90] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: representing knowledge about information systems. *ACM Transaction on Information Systems*, 8(4):325–362, October 1990.

[MBR01] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *27th VLDB*, 2001.

[MFT+03] George A. Miller, Christianne Fellbaum, Randee Tengi, Susanne Wolff, Pamela Wakefield, and Helen Langone. *WordNet: a lexical database for Engilish language*. Princeton University, http://www.cogsci.princeton.edu/ wn, 2003.

[MGMR02] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *18th ICDE*, 2002.

[MH03] Jayant Madhavan and Alon Y. Halevy. Composing mappings among data sources. In *29th VLDB, Berlin, Germany*, 2003.

[MHH00] Renee J. Miller, Laura M. Haas, and M. A. Hernandez. Schema mapping as query discovery. In *26th VLDB conference*, 2000.

[MKSI96] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: an approach for query processing in global information systems based on interoperation across preexisting ontologies. In *First IFCIS international conference on cooperative information systems*, 1996.

[MSZ01] Sheila McIlraith, Tran Cao Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, March/April 2001.

[NM01] Natalya F. Noy and Mark A. Musen. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In *Workshop on ontologies and information sharing at IJCAI-2001*, 2001.

[PVM+02] L. Popa, Y. Velegrakis, R. J. Miller, M. Hernandes, and R. Fagin. Translating Web Data. In *VLDB*, 2002.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB*, 10:334–350, 2001.

[She97] A. Sheth. Panel: Semantics in practical applications - what, when, and how? In *Database Applications Semantics, R. Meersman ed., Chapman and Hall published*, pages 601–610, 1997.

[Smi87] Brian Cantwell Smith. The Correspondence Continuum. Technical Report CSLI-87-71, Stanford University, 1987.

[SP94] S. Spaccapietra and C. Parent. View Integraion: A Step Forward in Solving Structural Conflicts. *TKDE*, 6(2):258–274, 1994.

[SW01] Barry Smith and Christopher Welty. Ontology: towards a new synthesis. In *FOIS'01, Ogunquit, Maine, USA*, 2001.

[SY80] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operators. *J. of ACM*, 27(4):633–655, 1980.

[vdM92] R. van der Meyden. *The complexity of querying indefinite information*. Ph.D. thesis, Rutgers University, 1992.

[WG01] Christopher Welty and Nicola Guarino. Supporting Ontological Analysis of Taxonomic Relationships. *Data and Knowledge Engineering*, 39(1):51–74, 2001.