Data Semantics: Semantic Encapsulation in Schemas

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

Alex Borgida
Department of Computer Science
Rutgers University
borgida@cs.rutgers.edu

John Mylopoulos
Department of Computer Science
University of Toronto
jm@cs.toronto.edu

15th December 2003

Abstract

The goal of this paper is to investigate the enduring problem of data semantics recurred in the context of data integration and the semantic web. The investigation is based on two principles, *correspondence continuum* and *semantic encapsulation*. The correspondence continuum postulates that data semantics amounts to a continuum of correspondences. The continuum starts from linguistic structures, midway across models with richer semantics, in some point, reaching the real world. The semantic encapsulation postulates that each model is equipped with semantics as being represented by the semantic correspondence to other modeling structures. The correspondences constitute the correspondence continuum which ultimately is grounded on a domain ontology. In contrast to the semantic web, the real world semantics supplied by the domain ontology will be reached through the chaining of correspondences. In doing this, we propose a formal correspondence language inspired by the study in database query languages. We also study the compositionality of correspondences anchored in a correspondence continuum. Furthermore, we examine the reasoning tasks and the corresponding algorithms for deriving semantic mapping between database schemas. Mapping generation which is the key issue in current data integration practices is carried out through correspondence composition and query reformulation.

1 Introduction

Investigating and capturing the meaning of data is a long-standing problem in all of data manipulation applications. Especially, in data integration and the semantic web the problem has been studied by many researchers. As a result, two loosely coupled paradigms have found their way into the two research areas. In data integration, a set of autonomous, heterogeneous, and distributed local data sources contain materialized contents for answering query globally. The key is to specify the mappings between local sources and a global schema. Much progress has been seen in the study of *schema integration and merging*, *schema matching*, and *schema mapping*. In examining the semantics of each source and schema, linguistic similarity and structural similarity are the primary vehicle

for detecting and guessing the semantic correspondences between elements of different schemas. Furthermore, data dependency theory also has been used in discovery of formal queries over source and target schemas.

In the course of data integration research, people have realized the role of ontology as the shared specification of conceptualization for an application domain. Many have attempted to introduce ontologies into the data integration paradigm. However, the overall paradigm has gained little improvement and enhancement, because most of the efforts merely substitute an ontology described in an existing data model for the global schema. Nevertheless, ontologies play a key role in the paradigm of the semantic web [BLHL01], which is an extension of the current Web. The goal of the semantic web is to give information well-defined meaning, better enabling computers and people to work in cooperation. As a recent descendant of knowledge representation, the paradigm of the semantic web makes use of knowledge representation languages and their rigorous semantic analysis results.

As much of the semantic web research treats the web as a knowledge base abounding with the definitions of meanings and relationships, it has been seen that disconnectivity between the semantic web world and most of today's data suppliers and applications emerges (also noted by [PSS02]). The another problem that the semantic web paradigm is facing with is that the assumption about the direct link from data to the authoritative ontology for a particular domain does not fit into today's data management practices well. Instead database designers and data management practitioners prefer to represent and manage data using a best fitted approach chosen from a diverse range, and data integration techniques restore links between most convenient data sets for interoperability (also noted by [HIMT03]).

The disconnectivity as well as the commonality between data integration practices and the semantic web proposals draws our attention again back to the key issue – capturing the meaning of data. Based on the insights into the paradigms examined above, we postulate that the problem of data semantics amounts to establishing and maintaining semantic correspondence from data to their intended subject matter. From modeling point of view, the subject matter could be the states of affairs of the real world, or it could be another modeling structure; however, in some point, some modeling structures should have the direct correspondence to the states of affairs of the real world. In a model world, we envisage a picture containing various modeling structures which are woven together by inter-correspondences. The picture gives rise to our two investigation principles: *correspondence continuum* and *semantic encapsulation*.

The principle of correspondence continuum enables one to carry out the semantical analysis of a model through its correspondences to other modeling structures. The continuum starts from intensional structures, midway across models with richer semantics, at some point, reaching the states of affairs of the real world that the original structure were genuinely about. The second principle, semantic encapsulation, postulates that each model should be equipped with its semantics which is encoded in the correspondence to its subject matter. Brian Smith in [Smi87] philosophically studies the correspondence continuum in the context of analyzing knowledge representation systems in general. We adopt the principle into the analysis of data semantics in information systems and leverage the techniques in a set of rich and well-understood database schema mediation languages. We show that the principles are well fitted into the paradigm of information exchange in a widely distributed environment such as the Web.

The specific contributions of the paper are the following:

- We propose the principles of *correspondence continuum* and *semantic encapsulation*, and we show a framework for capturing data semantics.
- We classify the representations of semantic correspondence into intensional and extensional ones. Different representations play distinct roles in the process of data exchange and integration. Leveraging the techniques of a set of rich and well-understood database schema mediation languages, we propose a correspondence language which is able to capture both intensional meaning and extensional meaning of correspondence. The formalism is the GAV-like in data integration notation, and the correspondence language is a conjunctive query involving selection, projection, and join. We believe both the formalisms and the language represent a wide variety of applications.
- We show that explicit representation of data semantics in terms of correspondences enables automatic reasoning upon it. The composition of correspondences makes two modeling structures reach their common ancestor. Techniques of answering and rewriting queries using views provide approaches for deriving semantic mapping between arbitrary two schemas with overlap. The derivation of semantic mapping shows that the explicit presentation of data semantics can also verify the correctness of other mapping discovery tools.

Before we proceed, we want to make a clear distinction between the study of data semantics in terms of semantic encapsulation and the study of the peer data management systems [HIST03, HIMT03]. In the peer data management systems (PDMS), each autonomous peer semantically maps its schema to other convenient peers. The study of PDMS is to characterize the query answering ability over the peer-to-peer infrastructure, leaving the problem of specifying the semantic mapping between peers untouched. In contrast, the study of data semantics aims to investigate and capture the meaning of data for each individual data source. It happens to show a resemblance between the correspondence continuum and the peer-to-peer infrastructure; however, they are two different problems. One connection between them is that the study of data semantics could be one basis of PDMS in characterizing the semantic mappings between peers. Of course, approaches to data semantics would be helpful for a wide variety of data management problems.

The rest of the paper is organized as follows. We start to investigate data semantics from a common database design scenario in Section 2, and we show the importance of correspondences between models. In Section 3 we propose principles of correspondence continuum and semantic encapsulation. We also present the correspondence language for characterizing both intensional and extensional meanings of a correspondence. In Section 4 we continue the investigation of correspondences in terms of reasoning tasks, and we show how to derive semantic mapping between arbitrary schemas. In Section 5 we present related work. Finally, In Section6 we propose future work directions and give conclusions.

2 Data Semantics in a Common Database Design 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.

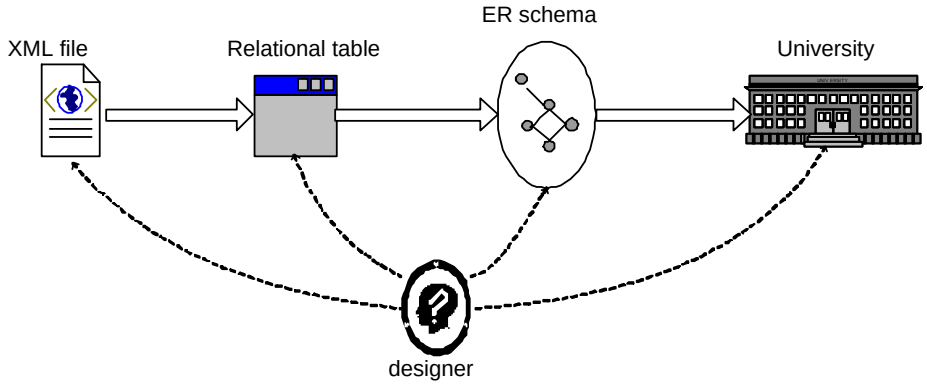

Figure 1: A university information systems design procedure

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.

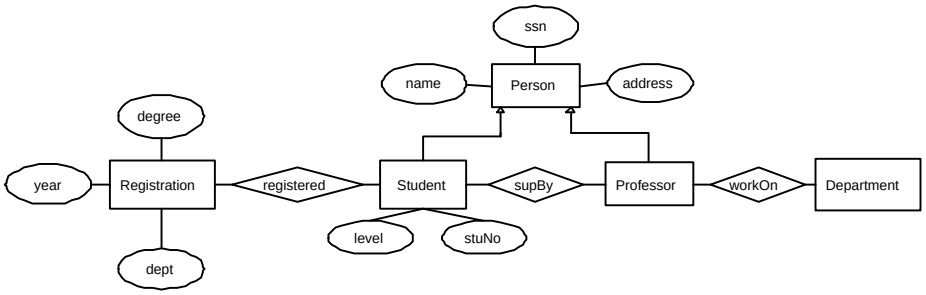

Figure 2: $\mathcal{E}_{ut}$: the Entity-Relationship diagram of the information systems of UT

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 $\mathcal{E}_{ut}$.

```
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

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

```
UTDB:
 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

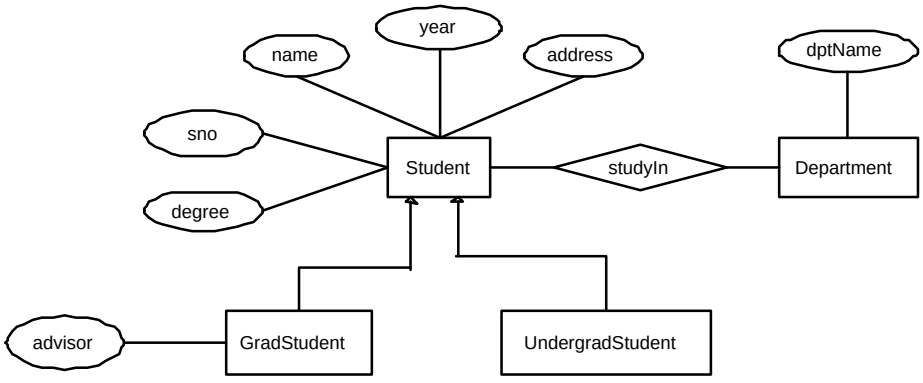

Figure 5: $\mathcal{E}_{uw}$: the Entity-Relationship diagram of the information systems of UW

```
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

```
UWDB:
    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

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 *UofW* 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 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.

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

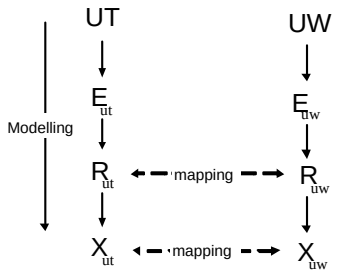

Figure 8: Lineal correspondence vs. semantic mapping.

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.

```
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⊓∃hasAdvisor.PROFESSOR.
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).
UNDERGRAD ⊑ UNIVERSITY-STUDENT⊓∀registerAt.(∀withDegree.UNDERGRAD-DEGREE).
```

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

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 the correspondence continuum and semantic encapsulation. Furthermore, we examine reasoning tasks revolving correspondences and any practical application of data semantics presentation.

3 The Correspondence Continuum and the Semantic Encapsulation

Schemas are useful for storing and querying data. As the product of the mental process of a designer, a schema represents the designer's intention for the structure of data as well as the meaning of data. As shown in the common scenario in Section 2, a schema is not out of vacuum but derived from the reflection on its intended subject matter. Data semantics is captured by principles of correspondence continuum and semantic encapsulation presented as follows.

3.1 Correspondence Continuum

The traditional model-theoretic analysis treats semantics as suggested in Figure 10 . A source syntactic domain as the set of elements for which semantics is to be given is identified. A semantic domain or *Domain of Discours*e, roughly to be the elements of the syntactic domain are about, is also identified. The semantic relation between domains, usually called the interpretation, is described extensionally, in the sense that particular elements of the syntactic domain are mapped, piece-wise, onto the corresponding particular elements of the semantic domain.

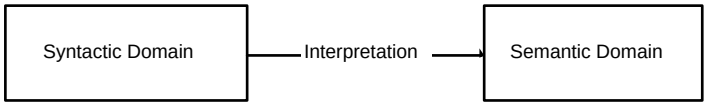

Figure 10: The model-theoretically semantic analysis

As in relational database, a relation name with n attributes is viewed as an n-ary predicate whose meaning is interpreted as a number of n-tuples in a particular domain. Tables are used to stored tuples; therefore, a relational database is a particular finite interpretation of the set of first-order formulas representing the relational database schema [Rei84].

Example 1. Consider the relation $\mathcal{R}_{ut}$*:Student(StuNo, Name, Address, Level, SupBy)* in Figure 3. If we have the following values in a particular university domain: { "B001", "B002", "B003", "Richard", "Anthony", "James", "10 King's College", "20 Dundas", "30 St. George", "Grad", "Underg", "Prof. Miller", null}, then the following table defines a particular finite interpretation of the predicate *Student(StuNo, Name, Address, Level, Supby):*

StuNo	*Name*	*Address*	*Level*	*SupBy*
"B001"	*"Richard"*	*"10 King's College"*	*"Grad"*	*"Prof. Miller"*
"B002"	*"Anthony"*	*"20 Dundas"*	*"Underg"*	*null*
"B003"	*"James"*	*"30 St. George"*	*"Underg"*	*null*

□

Generally speaking, the semantic domain can be any structure in terms of giving semantics to the syntactic domain. In the case where the elements of the semantic domain are themselves syntactic, bearing their own interpretation relation to another semantic domain, then the semantically denotational relation is taken to be *non-transitive*. For example, when the *name* attribute of the *student* relation in Example 1 is interpreted to the string "Richard", the string "Richard" has its own interpretation which is the name of the student who is Richard. However, we have not denoted the

name attribute as the name of the particular student, *Richard*. Rather it is denoted as a syntactic string, "Richard." [We may think the other way around.]

For integrating two databases, the first thing to do is identifying the similar relationships between database objects. For automatic similarity identification, the meaning of an object can be interpreted semantically by the correspondence to the schema element describing it, and the schema element in turn is semantically corresponding to its subject matter. As a result, the two-level model-theoretically denotational approach is inadequate to the complexities of a wide variety of semantic correspondences in reality. Specifically, there is a rich body of other relations each of which has its own right for the semantical analysis of the elements in a syntactic domain. Some of the relations called *genuine relations* in [Smi87] include:

1. The *specification* relation, e.g., the relation between a program and the process it engenders;
2. The *internalisation* and *externalisation* relations, e.g., the relations between linguistic expressions used by a system to communicate with its users, and the internal structure inside the system;
3. The *implementation* relation, e.g., the relation between the representation at one level, and a representation at other lower-level in terms of which it is implemented;
4. The *primary representation* relation, e.g., the relation between a model and the states of affairs in the world with which the system is concerned.

Particularly, the modelling relation can be composed as opposed to the denotational relation. For example, a photocopy of the picture of the parliament building is a model[1] of the picture, which in turn is a model of the real building. But it is harmless to say that the photocopy is also a model of the real building. In the common scenario example, we can say that the schema which is derived at one step is a model of the schema on the previous step as well as the model of the subject matter of the schema on the previous step and so on, as shown in Example 2.

Example 2. Consider the database design process of *UofT*: the Entity-Relationship schema $\mathcal{E}_{ut}$ in Figure 2 is a model of the subject matter of the *UofT* university; The relational schema $\mathcal{R}_{ut}$ in Figure 3 is a model of the Entity-Relational schema $\mathcal{E}_{ut}$; and the XML schema $\mathcal{X}_{ut}$ in Figure 4 is a model of the relational schema $\mathcal{R}_{ut}$. However, the modelling relations can be composed, i.e., the XML schema $\mathcal{X}_{ut}$ is also a model of the Entity-Relational schema $\mathcal{E}_{ut}$ as well as a model of the university as shown in Figure 9 .

□

In database design activities, modeling is the common correspondence between schemas. Having a chain of modeling correspondences from a schema element to the real world counterpart, we may connect a database object to the real world object by the composition of modeling correspondences. For example, by composition we may identify that the *name* attribute of the *student* relation in Example 1 models the collection of the names of the students in *UofT*. The string "Richard" is one of instances of *name* attribute; therefore, "Richard" is a student's name who is in *UofT*. Now

[1] It should be clear that this model is different from the model of a logic in the model-theoretic sense.

we need to present the correspondence continuum for a wide variety of semantic analysis situations which show various relations beyond modeling correspondence.

The moral is this: all the genuine relations mentioned before as well as other relations cover the whole spectrum of semantic relations. A given intentional structure – language, process, schema, model – is set in correspondence with one or more other structures, each of which is in turn set in correspondence with still others, at some point reaching the states of affairs in the world that the original structures were genuinely about. It is the structure that is called the "correspondence continuum" in [Smi87] as shown in Figure 11: a semantic soup in which to locate transitive and non-transitive linguistic relations, relations of modelling and encoding, implementation and specification, and the rest.

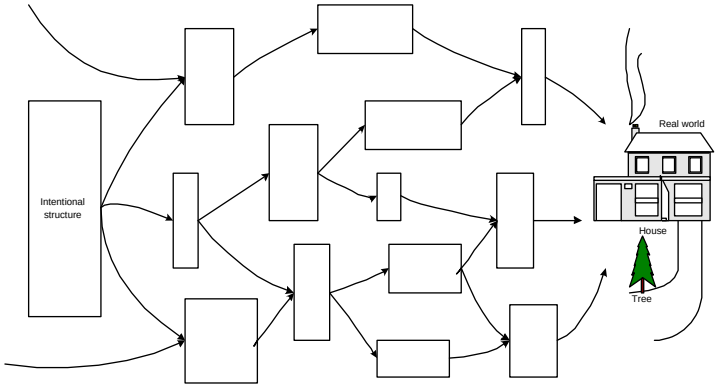

Figure 11: The correspondence continuum (semantic soup)

Apparently, there must be some ways to store the states of affairs about the real world in information systems. Thanks to the study of formal ontology. Now an ontology as the specification of conceptualization for a particular domain can be reached among domain experts as well as general users, and the ontology can be represented in formal languages and presented on the Web. Assuming the existence of domain ontologies, we will focus on the particular correspondence continuum consisting of specific semantic relations between database schemas as shown in Example 2. Next section will discuss a formal representation of the correspondence.

3.2 Formal Specification of Semantic Encapsulation in Database Schemas

The principle of *semantic encapsulation* states that each schema comes with its semantics presented in terms of the correspondence to other schema(s). The general structure of correspondence is depicted in Figure 12. The set of elements in a database *schema 1* have semantic correspondence to the set of elements of another database *schema 2* by taking certain circumstantial parameters into consideration. For machinery, formalisms are needed. First, we introduce the basic formalisms of domain ontology and various database schemas in Section 3.2.1. Next, we propose a formal correspondence language for the encapsulation of semantics in Section 3.2.2.

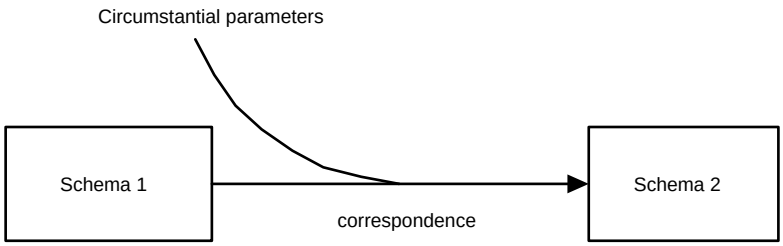

Figure 12: The general structure of correspondence

3.2.1 Basic Formalisms

We primarily focus on three types of structures: domain ontology, Enriched Entity-Relationship schema, and logical schemas including relational schema and XML schema. For each type of structure we introduce a formalism as follows.

- ***Domain Ontology.*** As shown in Figure 9, a domain ontology is described in a Description Logic which is fragments of First Order Logic and powerful enough to express a wide variety of domain knowledge. Concepts and roles are unary predicates and binary predicates, respectively. In addition, there exist decidable algorithms for reasoning the concept satisfiability and subsumption in a domain ontology. We notice that the proposed Web Ontology Language (OWL) has a formal semantics based on a Description Logic. It is appropriate to assume that a number of domain ontologies will be available described by OWL language with the proliferation of the Semantic Web.

- ***Enriched Entity Relationship Schema.*** The Entity-Relationship model is the standard conceptual modeling technique for database design. The following formalism allows ISA-relationship between entities and cardinality constraints on roles.

  Let $\mathcal{L}$ be an alphabet partitioned into a set $\mathcal{E}$ of entity symbols, a set $\mathcal{A}$ of attribute symbols, a set $\mathcal{U}$ of role symbols, a set $\mathcal{R}$ of relationship symbols, and a set $\mathcal{D}$ of domain symbols; each domain symbol D has an associated predefined basic domain dom.

  An Entity-Relationship schema is a 5-tuple $\mathcal{S} = (\mathcal{L}_\mathcal{S}, \preceq, att, rel, card)$, where $\mathcal{L}_\mathcal{S}$ is a finite subset of $\mathcal{L}$ and $\mathcal{L}_\mathcal{S} = \mathcal{E}_\mathcal{S} \cup \mathcal{A}_\mathcal{S} \cup \mathcal{U}_\mathcal{S} \cup \mathcal{R}_\mathcal{S} \cup \mathcal{D}_\mathcal{S}$ such that $\mathcal{E}_\mathcal{S} \in \mathcal{E}$, $\mathcal{A}_\mathcal{S} \in \mathcal{A}$, $\mathcal{U}_\mathcal{S} \in \mathcal{U}$, $\mathcal{R}_\mathcal{S} \in \mathcal{R}$, and $\mathcal{D}_\mathcal{S} \in \mathcal{D}$.

  - $\preceq \subseteq \mathcal{E}_\mathcal{S} \times \mathcal{E}_\mathcal{S}$ represents the ISA-relationship between entities.
  - $att : \mathcal{E}_\mathcal{S} \rightarrow \{< A : D > |\ A \in \mathcal{A}_\mathcal{S},\ D \in \mathcal{D}_\mathcal{S}\}$ is a function mapping an entity to a set of attributes.
  - $rel : \mathcal{R}_\mathcal{S} \rightarrow \{< U_1 : E_1, ..., U_k : E_k > |\ U_i \in \mathcal{U}_\mathcal{S}, E_i \in \mathcal{E}_\mathcal{S}, i = 1, .., k\}$ is a function mapping an relation symbol to a tuple of entities with role labels.
  - $card : \mathcal{E}_\mathcal{S} \times \mathcal{R}_\mathcal{S} \times \mathcal{U}_\mathcal{S} \rightarrow \mathbb{N}_0 \times (\mathbb{N}_0 \cup \{\infty\})$ ($\mathbb{N}_0$ is the set of natural numbers) is a cardinality function with the minimum cardinality as its first component and the maximum cardinality as its second component.

In a fragment of first order logic, symbols in $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{D}$ are unary predicates, and symbols in $\mathcal{A}$ and $\mathcal{U}$ are binary predicates. The $\preceq$ relation is interpreted as: $\forall x(E_1(x) \rightarrow E_2(x))$ if $(E_1, E_2) \in \preceq$. Without labels in $\mathcal{U}$, a relationship symbol in $\mathcal{R}$ can be viewed as an n-ary predicate if there are n entities participating the relationship.

- ***Logical Schemas.*** At the logical level, relational databases and XML files are used to describe the structures of data. Formally, a relation with n attributes is viewed as a n-ary predicate; and the tree-like XML structure is formalized in nested relational model, as shown in the following example.

Example 3. For the XML schema $\mathcal{X}_{ut}$ as shown in Figure 4, the logical associations among schema elements will be expressed in the form of logical relation as shown below:

(s.student.level, s.student.StuNo, s.student.Name, s.student.Address, r.Registration.Degree, r.Registration.Department, r.Registration.Year | s∈UTstudentDB.students, r∈s.registrations)
represents registered students who have no supervisors, and

(s.student.level, s.student.StuNo, s.student.Name, s.student.Address, r.Registration.Degree, r.Registration.Department, r.Registration.Year, p.SupBy.name | s∈UTstudentDB.students, r∈s.registrations, p∈s.Professors)
represents registered students who have supervisors.

□

In the next section, we will establish correspondences between the three types of formalisms introduced above.

3.2.2 A Formal Correspondence Language

Notation Convention. Given a structure $\mathcal{S} = (\Omega, \Sigma)$ where Ω is an alphabet and Σ is a set of expressions over Ω in a language $\mathcal{L}_{\mathcal{S}}$, $\mathcal{S}$ may be a type of ontology, conceptual schema, or logical schema, we will refer to the elements of Ω as $\mathcal{S} : E$ for an element $E \in \Omega$.

Semantic Correspondence. Consider the entity $\mathcal{E}_{ut}$:*Student* in the Entity-Relationship schema $\mathcal{E}_{ut}$ in Figure 2, semantically, it models the students who are officially registered in the university *UofT* and pursuing some kinds of degrees. Having an academic ontology $\mathcal{O}$ (Figure 9) abstracting concepts and relationships among concepts for a class of academic organization, we can specify the semantics of $\mathcal{E}_{ut}$:*Student* in terms of a correspondence statement relating to an expression over $\mathcal{O}$, as follows:

$\mathcal{E}_{ut}$*:Student(x)* ← $\mathcal{O}$*:UNIVERSITY-STUDENT(x),*$\mathcal{O}$*:REGISTRATION(y),*$\mathcal{O}$ *:UNIVERSITY(z),*
$\mathcal{O}$*:registerAt(x,y),*$\mathcal{O}$*:ofUniv(y, z),*$\mathcal{O}$ *:hasName(z, "UofT").*

Intensional vs. Extensional Correspondence. Careful analysis reveals that elements of one structure may correspond to elements of another structure because they may have either similar sense or same extension, or both. For example, both the Entity-Relationship schema $\mathcal{E}_{ut}$ and the relational schema $\mathcal{R}_{ut}$ model the same application domain. If there is a correspondence relating the

element $\mathcal{R}_{ut}$:*Student(StuNo, Name, Address, Level, SupBy)* to the elements $\mathcal{E}_{ut}$:*Student* , $\mathcal{E}_{ut}$*:name*, $\mathcal{E}_{ut}$*:stuNo*, $\mathcal{E}_{ut}$:*address*, $\mathcal{E}_{ut}$*:level* and $\mathcal{E}_{ut}$*:supBy* in Figure 2, that means the correspondence relates them at the extensional level as well as the intensional level. However, in some cases one structure corresponds to another structure probably they have same sense, but they may have different extensions. For example, there is another university, *UofD*, designing its information system by totally borrowing the schemas of *UofT*. Let $\mathcal{R}_{ud}$ be its relational schema for storing the operational data. Both relational schemas $\mathcal{R}_{ud}$ and $\mathcal{R}_{ut}$ have same structure and there is a one-to-one correspondence between them. However, the two databases have different data because there is no extensional data overlap between the two universities. In this case, we say that $\mathcal{R}_{ud}$ correspond to $\mathcal{R}_{ut}$ intensionally. By this analysis, we propose our formal correspondence language containing two types of correspondence statements: intensional correspondence and extensional correspondence.

Formal Correspondence Language. Given two schemas $\mathcal{S}$ and $\mathcal{T}$, where $\mathcal{S}$ is expressed in a language $\mathcal{L}_{\mathcal{S}}$ over an alphabet $\mathcal{A}_{\mathcal{S}}$, and $\mathcal{T}$ is expressed in a language $\mathcal{L}_{\mathcal{T}}$ over an alphabet $\mathcal{A}_{\mathcal{T}}$, then we use the following formal correspondence statement to associate an element of $\mathcal{S}$ with an expression over elements of $\mathcal{T}$, if, intuitively, they have same sense, i.e., intensional correspondence :

$$\mathcal{S} : E(\vec{x}) \leftarrow_{int} \mathcal{T} : Q(\vec{x}, \vec{y}). \tag{1}$$

And, we use the following formal correspondence statement to specify that an element of $\mathcal{S}$ has extensional correspondence with an expression over elements of $\mathcal{T}$:

$$\mathcal{S} : E(\vec{x}) \leftarrow_{ext} \mathcal{T} : Q(\vec{x}, \vec{y}). \tag{2}$$

Where the head $E(\vec{x})$ is an element which represents a certain type of logical association of a set of atomic elements in $\mathcal{S}$, such as a relation in a relational schema and a subtree of a tree-like XML schema.

Example 4. The relations *Student(StuNo, Name, Address, Level, SupBy)* and *Registration(StuNo, Year, Degree, Dept)* can be heads of some correspondence statements for the relational schema $\mathcal{R}_{ut}$; the form of subtree (or logical relation in literature such as [PVM$^+$02]) :

(s.student.level, s.student.StuNo, s.student.Name, s.student.Address, r.Registration.Degree, r.Registration.Department, r.Registration.Year | s∈UTstudentDB.students, r∈s.registrations)
can be the head of a correspondence statement for the XML schema $\mathcal{X}_{ut}$.

□

$\mathcal{T} : Q(\vec{x}, \vec{y})$ in the correspondence statement (2) is a query expression in a query language $\mathcal{L}_q$ over schema $\mathcal{T}$.

Example 5. For the Entity-Relationship model and the relational model, $Q(\vec{x}, \vec{y})$ is union of a set of formulas each of which has the form

$$conj(\vec{x}, \vec{y})$$

where each $conj(\vec{x}, \vec{y})$ is a conjunction of atoms, and $\vec{x}$, $\vec{y}$ are all the variables appearing in the conjunct. Each atom is a predicate with arity designating an entity, a relation, an attribute as a binary predicate, or declaring the domains of variables and the value correspondence between domains. It may use predicates other than those of $\mathcal{T}$.

□

Leveraging Data Integration Formalisms. There are three formalisms proposed in literature for specifying a data integration system: local-as-view (LAV), global-as-view (GAV), and global-local-as-view (GLAV) [Len02]. In LAV, the mapping formula, $q_{\mathcal{S}} \rightarrow q_{\mathcal{G}}$, relates a single element $q_{\mathcal{S}}$ of a source schema $\mathcal{S}$ to a query expression $q_{\mathcal{G}}$ over the global schema $\mathcal{G}$; in GAV, $q_{\mathcal{S}}$ is a query expression, and $q_{\mathcal{G}}$ is a single element; and in GLAV, both $q_{\mathcal{S}}$ and $q_{\mathcal{G}}$ are query expressions. For our correspondence statements (1) and (2) above, from the standpoint of a schema and the perspective of capturing its semantics in terms of the correspondence continuum, the statements are equivalent to the GAV formalism; therefore, the correspondence composition can be simply carried out by unfolding operations which will be seen in later section. However, a reasoning task making use of the semantic encapsulation to derive the semantic mapping between arbitrary schemas with overlap will need to leverage query answering techniques developed in LAV formalism. This is the main topic of the section to follow.

The Semantics of the Correspondence Statement. For both the intensional and the extensional correspondence statements, their denotational semantics can be specified in the same manner. That is, given a database $\mathcal{D}$ that satisfies $\mathcal{T}$, a correspondence statement,

$$\mathcal{S} : E(\vec{x}) \leftarrow \mathcal{T} : Q(\vec{x}, \vec{y}),\ where\, \vec{x} = \{x_1,\ ...\ , x_n\}\ and\ \leftarrow \in \{\leftarrow_{int}, \leftarrow_{ext}\},$$

is interpreted as the set of n-tuples $(a_1, a_2,\ ...\ , a_n)$ with each a_i an object of the database $\mathcal{D}$, such that, when substituting each a_i for x_i, then formula

$$\exists y.\mathcal{T} : conj(\vec{x}, \vec{y})$$

evaluates to true.

The semantics is straightforward to the extensional correspondence, because both schema $\mathcal{S}$ and schema $\mathcal{T}$ are intended to model the same subject matter; therefore, a single database $\mathcal{D}$ is used to specify the semantics. However, $\mathcal{S}$ and $\mathcal{T}$ may model different subject matters in the intensional correspondence. So how can a single database gives the semantics to an intensional correspondence statement? The answer is that the intensional correspondence statement $\mathcal{S} : E(x) \leftarrow_{int} \mathcal{T} : Q(x, y)$ means if $\mathcal{S}$ and $\mathcal{T}$ model the same subject matter, it will behave like the extensional correspondence. Hence, we use two distinct symbols to differentiate the the two types of correspondences. Following examples demonstrate the specifications of the semantics of schemas in terms of correspondence statements for the common scenario example in Section 2.

Example 6. The following statements specify the extensional and intensional correspondences from the student element in $\mathcal{E}_{ut}$ to the expressions over elements of $\mathcal{O}$:

$\mathcal{E}_{ut}$*:Student(x)* $\leftarrow_{ext}$	*$\mathcal{O}$:UNIVERSITY-STUDENT(x),$\mathcal{O}$:REGISTRATION(y),$\mathcal{O}$:UNIVERSITY(z),*
	$\mathcal{O}$:registerAt(x,y),$\mathcal{O}$:ofUniv(y, z),$\mathcal{O}$:hasName(z, "UofT").
$\mathcal{E}_{ut}$*:Student(x)* $\leftarrow_{int}$	*$\mathcal{O}$:UNIVERSITY-STUDENT(x).*

□

Example 7. The following statements specify how the student relation in $\mathcal{R}_{ut}$ corresponds to $\mathcal{E}_{ut}$, extensionally and intensionally:

$\mathcal{R}_{ut}$:Student(StuNo, Name, Address. Level, SupBy) $\leftarrow_{ext}$ / $\leftarrow_{int}$
$\mathcal{E}_{ut}$:Student(x), $\mathcal{E}_{ut}$:name(x, Name), $\mathcal{E}_{ut}$:address(x, Address), $\mathcal{E}_{ut}$:stuNo(x, StuNo), $\mathcal{E}_{ut}$:level(x, Level), $\mathcal{E}_{ut}$:supBy(x, y), $\mathcal{E}_{ut}$:Professor(y), $\mathcal{E}_{ut}$:name(y, SupBy), String(StuNo), String(Name), String(Address), String(level), String(SupBy), Represent([StuNo], x).

Where $\leftarrow_{ext}$ / $\leftarrow_{int}$ means either one can be used, and the predicate String() represents the string simple domain value. The predicate Represent([StuNo],x) specifies that the tuple [StuNo] will be used as the identifier of the entity x in the Student relation of $\mathcal{R}_{ut}$.

□

Example 8. Similarly, we can specify the correspondence between the Entity-Relational schema $\mathcal{E}_{uw}$ and the ontology $\mathcal{O}$ as well as the correspondence between $\mathcal{R}_{uw}$ and $\mathcal{E}_{uw}$ as the following statements:

$\mathcal{E}_{uw}$:GradStudent(x) $\leftarrow_{ext}$ *$\mathcal{O}$:GRAD(x),$\mathcal{O}$:REGISTRATION(y),$\mathcal{O}$:UNIVERSITY(z), $\mathcal{O}$:registerAt(x,y),$\mathcal{O}$:ofUniv(y, z),$\mathcal{O}$:hasName(z, "UofW").*

$\mathcal{E}_{uw}$:GradStudent(x) $\leftarrow_{int}$ *$\mathcal{O}$:GRAD(x).*

$\mathcal{E}_{uw}$:UndergradStudent(x) $\leftarrow_{ext}$ *$\mathcal{O}$:UNDERGRAD(x),$\mathcal{O}$:REGISTRATION(y),$\mathcal{O}$:UNIVERSITY(z), $\mathcal{O}$:registerAt(x,y),$\mathcal{O}$:ofUniv(y, z),$\mathcal{O}$:hasName(z, "UofW").*

$\mathcal{E}_{uw}$:UndergradStudent(x) $\leftarrow_{int}$ *$\mathcal{O}$:UNDERGRAD(x).*

$\mathcal{R}_{uw}$:UndergradStudent(sno, name, address, year, degree) $\leftarrow_{ext}$ / $\leftarrow_{int}$
$\mathcal{E}_{uw}$:UndergradStudent(x), $\mathcal{E}_{uw}$:sno(x, sno), $\mathcal{E}_{uw}$:name(x, name), $\mathcal{E}_{uw}$:address(x, address), $\mathcal{E}_{uw}$:year(x, year), $\mathcal{E}_{uw}$:degree(x, degree), String(sno), String(name), String(address), Date(year), USFormat(year), String(degree), Represent([sno],x).

□

Example 9. If there is a third university, *UofD*, which designs its student information systems by borrowing the relational schema of the university UofT for storing the operational data, then there will be an one-to-one intensional correspondence between the relational schema of *UofD*, $\mathcal{R}_{ud}$, and the relational schema of *UofT*, $\mathcal{R}_{ut}$, such as,

$\mathcal{R}_{ud}$:Student(StuNo, Name, Address, Level, SupBy) $\leftarrow_{int}$
$\mathcal{R}_{ut}$:Student(StuNo, Name, Address, Level, SupBy).

However, we cannot specify their extensional correspondence.

□

Example 10. In order to specify the correspondence between the XML schema $\mathcal{X}_{ut}$ and the relational schema $\mathcal{R}_{ut}$ in the framework of the correspondence language, we will use the notation like *(s.Student.Name[$Name] | s∈UTDB.Students)* to indicate that the *Name* variable appearing at the

tree position following the path *s.Student.Name*, where *s*∈*UTDB.Students*, see Figure 4. The "Students" in *UTDB.Students* is a virtual set name indicating a set of students in the XML data model , each of which is a student data type which is a type of sub-tree. Then, the following statement specifies the correspondence between $\mathcal{X}_{ut}$ and $\mathcal{R}_{ut}$:

$\mathcal{X}_{ut}$:(s.Student.level[$level], s.Student.StuNo[$StuNo], s.Student.Name[$Name], s.Student.Address[$Address], r.Registration.Degree[$Degree], r.Registration.Department[$Department], r.Registration.Year[$Year]|s∈UTDB.Students, r∈s.Registrations) $\leftarrow_{ext}$/ $\leftarrow_{int}$
$\mathcal{R}_{ut}$:Student(StuNo, Name, Address, Level), $\mathcal{R}_{ut}$:Registration(StuNo, Year, Degree, Department).

□

4 Reasoning about Correspondences

In this section we will explore the opportunities provided by the correspondence continuum for deducing implicit semantic mapping between two schemas with overlaps. As we know, semantic mappings play a key role in data integration systems, and specifying semantic mappings is heuristic and user-intensive in most of current applications. The reasoning task involves two steps of operations: correspondence composition and mapping generation.

4.1 Correspondence Composition

For simplicity of exposition, we will use the three types of structures in the common scenario example in Section 2 for the investigation of correspondence composition, namely, the relational schema $\mathcal{R}_{ut}$, the entity-relational schema $\mathcal{E}_{ut}$, and the domain ontology $\mathcal{O}$. As shown in Section 3.2.2, several correspondence statements which specify the extensional and intensional correspondence between $\mathcal{R}_{ut}$ and $\mathcal{E}_{ut}$, and between $\mathcal{E}_{ut}$ and $\mathcal{O}$ have been given in Example 6 and Example 7. We want to compose the correspondences $\mathcal{R}_{ut} \leftarrow \mathcal{E}_{ut}$ and $\mathcal{E}_{ut} \leftarrow \mathcal{O}$ to obtain a direct correspondence $\mathcal{R}_{ut} \leftarrow \mathcal{O}$ to satisfy our reasoning purpose, where $\leftarrow$ can be either the intensional correspondence symbol $\leftarrow_{int}$ along all the composition path or the extensional correspondence symbol $\leftarrow_{ext}$ along all the composition path. The notation $\mathcal{R} \leftarrow \mathcal{E}$ represents a set of correspondence statements. Formally, we can define the correspondence composition as follows.

Definition 1. *(Correspondence Composition)***.** Given three structures $\mathcal{R}$, $\mathcal{E}$, and $\mathcal{O}$, the correspondence $\mathcal{R} \leftarrow \mathcal{O}$ is the composition of the correspondences $\mathcal{R} \leftarrow \mathcal{E}$ and $\mathcal{E} \leftarrow \mathcal{O}$, if the correspondence symbols are consistent along the correspondence path, i.e., only an intensional correspondence composes with another intensional correspondence resulting in a intensional correspondence, and similar for extensional correspondence; and given a database $\mathcal{D}$ that satisfies $\mathcal{O}$, the resulting database $\mathcal{D}'$ computed by the direct correspondence coincides with the resulting database $\mathcal{D}''$ computed by the two separate correspondences, according to the semantics of correspondence.□

Ignoring the distinction of intensional and extensional symbols temporarily and from the perspective of GAV formalism, we observe that the correspondence composition amounts to a query unfolding operation. Informally, we can describe the composition algorithm as follows.

Algorithm of Composition. *compose(Cor_1,Cor_2)***:** Let Cor_1 be a finite set of correspondence statements representing the correspondence $\mathcal{R} \leftarrow \mathcal{E}$, let Cor_2 be a finite set of correspondence

statements representing the correspondence $\mathcal{E} \leftarrow \mathcal{O}$, for each statement $\mathcal{R} : E(x) \leftarrow \mathcal{E} : Q(x,y)\} \in Cor_1$, for each $\mathcal{E}$ element $\mathcal{E} : P(z)$ appearing in $\mathcal{E} : Q(x,y)$, where $z \subset \{x,y\}$, replace $P(z)$ with the body of the correspondence statement $\mathcal{E} : P(z) \leftarrow \mathcal{O} : Q'(z,u)\} \in Cor_2$. After substitution, redundancy can be removed accordingly.□

The following proposition follows from the semantics of correspondence statement.

Proposition 1. The algorithm *compose(Cor_1, Cor_2)* terminates and computes the finite set of correspondence statements $\mathcal{R} \leftarrow \mathcal{O}$ representing the composition of the correspondences $\mathcal{R} \leftarrow \mathcal{E}$ and $\mathcal{E} \leftarrow \mathcal{O}$ as defined in Definition 1.□

A simple counting shows that the algorithm has polynomial time complexity in terms of the number of elements in the structures. The following example shows some composition results for the common scenario example.

Example 11. *(Composition of $\mathcal{R}_{ut} \leftarrow_{int(ext)} \mathcal{E}_{ut}$ and $\mathcal{E}_{ut} \leftarrow_{int(ext)} \mathcal{O}$ in Example 6 and Example 7).* Aside from the statements in Example 6, we need other correspondence statements of $\mathcal{E}_{ut} \leftarrow_{int(ext)} \mathcal{O}$ for composition. For brevity's sake, we only give the correspondences about the $\mathcal{E}_{ut}$:*name* element.

$\mathcal{E}_{ut}$:name(x,y) $\leftarrow_{ext}$ *$\mathcal{O}$:UNIVERSITY-STUDENT(x),$\mathcal{O}$:REGISTRATION(z),$\mathcal{O}$:UNIVERSITY(w),*
$\mathcal{O}$:registerAt(x,z),$\mathcal{O}$:ofUniv(z,w), $\mathcal{O}$:hasName(w, "UofT"),$\mathcal{O}$:hasName(x,u),
$\mathcal{O}$:PERSON-NAME(u), $\mathcal{O}$:hasFirstName(u,y1),$\mathcal{O}$:hasLastName(u,y2), concat(y1,y2,y).

$\mathcal{E}_{ut}$:name(x,y) $\leftarrow_{int}$ *$\mathcal{O}$:UNIVERSITY-STUDENT(x),$\mathcal{O}$:hasName(x,z),$\mathcal{O}$:PERSON-NAME(z),*
$\mathcal{O}$:hasFirstName(z,y1),$\mathcal{O}$:hasLastName(z,y2), concat(y1,y2,y).

The following statements are the resulting correspondence statements after composing the *Student* relation in $\mathcal{R}_{ut}$ with corresponding elements in $\mathcal{E}_{ut}$.

If the tuple is about an undergraduate student, extensionally:

$\mathcal{R}_{ut}$:Student(StuNo, Name, Address, Level, SupBy)$\leftarrow_{ext}$
$\mathcal{O}$:UNDERGRAD(x),$\mathcal{O}$:REGISTRATION(y),$\mathcal{O}$:UNIVERSITY(z),$\mathcal{O}$:registerAt(z,y),$\mathcal{O}$:ofUniv(y,z),
$\mathcal{O}$:hasName(z, "UofT"),$\mathcal{O}$:hasStuno(x,StuNo), $\mathcal{O}$:hasName(x,u),$\mathcal{O}$:PERSON-NAME(u),$\mathcal{O}$:hasFirstName(u,y1),
$\mathcal{O}$:hasLastName(u,y2), concat(y1,y2,Name), $\mathcal{O}$:hasAddress(x, Address), Level="Undergrad", $\mathcal{O}$:hasAdvisor(x,v),
$\mathcal{O}$:PROFESSOR(v),$\mathcal{O}$:hasName(v,SupBy), String(StuNo), String(Name), String(Address), String(SupBy),
Represent([StuNo],x).

If the tuple is about a graduate student, extensionally:

$\mathcal{R}_{ut}$:Student(StuNo, Name, Address, Level, SupBy)$\leftarrow_{ext}$
$\mathcal{O}$:GRAD(x),$\mathcal{O}$:REGISTRATION(y),$\mathcal{O}$:UNIVERSITY(z),$\mathcal{O}$:registerAt(z,y),$\mathcal{O}$:ofUniv(y,z),
$\mathcal{O}$:hasName(z, "UofT"),$\mathcal{O}$:hasStuno(x,StuNo), $\mathcal{O}$:hasName(x,u),$\mathcal{O}$:PERSON-NAME(u),$\mathcal{O}$:hasFirstName(u,y1),
$\mathcal{O}$:hasLastName(u,y2), concat(y1,y2,Name), $\mathcal{O}$:hasAddress(x, Address), Level="Grad", $\mathcal{O}$:hasAdvisor(x,v),
$\mathcal{O}$:PROFESSOR(v),$\mathcal{O}$:hasName(v,SupBy), String(StuNo), String(Name), String(Address), String(SupBy),
Represent([StuNo],x).

If the tuple is about an undergraduate student, intensionally:

$\mathcal{R}_{ut}$:*Student(StuNo, Name, Address, Level, SupBy)*$\leftarrow_{int}$
$\mathcal{O}$:*UNDERGRAD(x)*,$\mathcal{O}$:*hasStuno(x,StuNo)*, $\mathcal{O}$:*hasName(x,u)*,$\mathcal{O}$:*PERSON-NAME(u)*,$\mathcal{O}$:*hasFirstName(u,y1)*,
$\mathcal{O}$:*hasLastName(u,y2)*, *concat(y1,y2,Name)*, $\mathcal{O}$:*hasAddress(x, Address)*, *Level="Undergrad"*, $\mathcal{O}$:*hasAdvisor(x,v)*,
$\mathcal{O}$:*PROFESSOR(v)*,$\mathcal{O}$:*hasName(v,SupBy)*, *String(StuNo)*, *String(Name)*, *String(Address)*, *String(SupBy)*,
Represent([StuNo],x).

If the tuple is about a graduate student, intensionally:

$\mathcal{R}_{ut}$:*Student(StuNo, Name, Address, Level, SupBy)*$\leftarrow_{int}$
$\mathcal{O}$:*GRAD(x)*,$\mathcal{O}$:*hasStuno(x,StuNo)*, $\mathcal{O}$:*hasName(x,u)*,$\mathcal{O}$:*PERSON-NAME(u)*,$\mathcal{O}$:*hasFirstName(u,y1)*,
$\mathcal{O}$:*hasLastName(u,y2)*, *concat(y1,y2,Name)*, $\mathcal{O}$:*hasAddress(x, Address)*, *Level="Grad"*, $\mathcal{O}$:*hasAdvisor(x,v)*,
$\mathcal{O}$:*PROFESSOR(v)*,$\mathcal{O}$:*hasName(v,SupBy)*, *String(StuNo)*, *String(Name)*, *String(Address)*, *String(SupBy)*,
Represent([StuNo],x).

□

Example 11 shows that we can view the extensional correspondences as specialized intensional correspondences in some ways. The bodies of the correspondence statements precisely tell the meaning of the head element. In next section, we will derive semantic mapping between arbitrary structures with semantic overlap by exploring their semantic correspondences to a common structure.

4.2 Semantic Mapping Generation

Specification of semantic mappings between data sources or peers, or between sources and mediated schemas is the key component of a wide variety of data sharing and integration systems. With the terrain of the data sharing and integration systems being expanded to the WWW, there is a succession of research and commercial tools for automatic or semi-automatic specification of semantic mappings. The most prominent one is the Clio [PVM$^+$02] tool which exploits integrity constraints and internal structures of database schemas to discover mappings between a source schema and a target schema. The resulting mappings are elicited by a set of user-defined value correspondences; therefore, it is intrinsically semi-automatic. Such a tool is, however, inadequate to establish correct mappings, even though a user completely understand the semantics of schemas in some cases as shown in the example below.

Example 12. Suppose a source schema $\mathcal{S}$ have one relation $p(x, y)$ and a target schema $\mathcal{T}$ also have one relation $q(x, y)$. The semantics of the schemas can be specified using the following formulas:

$$\mathcal{S} : p(x, y) \leftarrow \mathcal{O} : E(x, y). \qquad (12.1)$$
$$\mathcal{T} : q(x, y) \leftarrow \mathcal{O} : E(x, z), \mathcal{O} : E(z, y). \qquad (12.2)$$

That is, $p(x, y)$ means there is an edge between x and y; $q(x, y)$ means there is a path with length of 2 from x to y. The correct mapping should be $\forall(\mathcal{S} : p(x, z), \mathcal{S} : p(z, y))\exists q(x, y)$. Clio fails in this case. However, if the semantic correspondences like (12.1), (12.2) have been maintained, the correct mapping would be produced by an appropriate reasoning algorithm.

□

When the semantics of schemas is explicitly represented in terms of correspondences, and correspondence composition algorithm enables two schemas be mediated by a domain ontology, we are able to leverage the rich body of existing , scalable, and practical techniques developed for view-based queries answering in the database community. Especially, the sophisticated mapping composition algorithm in terms of a class of CQ_k queries in [MH03] provides right solution for our semantic mapping generation problem in terms of correspondence continuum.

We begin with the results of mapping composition developed in [MH03]. In the global-local-as-view (GLAV) formalism between data sources for a data sharing system, semantic mapping between two data sources A and B is specified by a set of mapping formulas, each of the form $Q_A(X) \subseteq Q_B(X)$, where Q_A and Q_B are conjunctive queries over schema of A, R_A, and schema of B, R_B , respectively. The mapping is denoted as $M_{A \to B}$. Given three data sources A, B and C, and two mappings $M_{A \to B}$ and $M_{B \to C}$, the composition of $M_{A \to B}$ and $M_{B \to C}$ is a direct mapping $M_{A \to C}$ such that the certain answers for any query Q over C computed through $M_{A \to C}$ are the same certain answers for the query Q over C computed through the two separate mappings w.r.t. all databases D for A. We can easily find the analogy between this mapping composition problem and our semantic mapping generation problem for ontology-mediated schemas. [MH03] shows that for relatively simple mappings involving two LAVs which treats B as the global schema, and treats A and C as two local schema, the composed mappings may be an infinite set of GLAV formulas.

Fortunately, there exists an algorithm that encodes an infinite number of GLAV formulas in the composition using finite graph structure. The key to the algorithm is to associate *residues* with formulas in the composition and build the composition incrementally. For the class of CQ_k queries with a finite set of residues, the algorithm is guaranteed to terminate and to encode the entire composition. Finally, [MH03] shows that the decision problem of whether a finite set of formulas is a composition is Π_2^p w.r.t. the set of conjunctive queries.

In light of the results of mapping composition, we are able to automatically generate intensional and extensional semantic mappings between arbitrary two schemas with overlap. First, for simplicity's sake, we make two assumptions: (i) the arbitrary two schemas are about same application domain and there exists one domain ontology as the consensus; (ii) each schema corresponds to only one other schema/model in the correspondence continuum. The assumption (ii) allows each schema ultimately be mediated by the domain ontology through correspondence composition. Second, we compose the correspondence which links a schema to its corresponding one and the correspondence which links the corresponding one to the next corresponding one and so on, until reaching the domain ontology. Of course, only an intensional correspondence can be composed with another intensional correspondence, and the same rule applies to extensional correspondences. Third, when both schemas are corresponding to the domain ontology, the mapping composition techniques developed in [MH03] can be applied after taking consideration about the specificity of our correspondence language as follows.

- **Dealing with the Domain Value Predicates**. When specifying the correspondence between schemas, we may introduce domain value predicates occasionally in the body of a correspondence statement to indicate the permissible values for an atomic element. For example, *String(StuNo)* indicates that the atomic element *StuNo* has value in the string domain. We assume there is a finite set of domain symbols $D = \{D_1, D_2, ..., D_m\}$, and there is a finite set of domain relation symbols $R = \{r_{ij}\}$, each of the relation symbols specifying how the elements in domain D_i relates to the elements in domain D_j. Furthermore, we use the

notation $X \downarrow_{r_{ij}}$ to indicate a new variable with domain D_j which relates to the variable X with domain D_i by domain relation r_{ij}. Having that, we can use the new variable notation to replace the original variable and change the domain value predicates back and forth during the query rewriting phase. The final mapping formulas will have the new variable notations embedded.

Example 13. Suppose after a series of composition operations, we have the following two correspondence statements:

$\mathcal{R}_{ut}$:*Grade(StuNo, Grade)*$\leftarrow_{int}$
$\mathcal{O}$:*UNIVERSITY-STUDENT(x),* $\mathcal{O}$:*hasStuno(x,StuNo),* $\mathcal{O}$:*hasGrade(x, Grade), Letter(Grade).*

$\mathcal{R}_{uw}$:*Mark(stuno, mark)*$\leftarrow_{int}$
$\mathcal{O}$:*UNIVERSITY-STUDENT(x),* $\mathcal{O}$:*hasStuno(x,stuno),* $\mathcal{O}$:*hasGrade(x, mark), Percentage(mark).*

Using domain relation symbol and the new variable notation, we can rewrite the second formula as follows:

$\mathcal{R}_{uw}$:*Mark(stuno, mark*$\downarrow_{r_{Percentage,Letter}}$*)*$\leftarrow_{int}$
$\mathcal{O}$:*UNIVERSITY-STUDENT(x),* $\mathcal{O}$:*hasStuno(x,stuno),* $\mathcal{O}$:*hasGrade(x, mark*$\downarrow_{r_{Percentage,Letter}}$*),*
Letter(mark$\downarrow_{r_{Percentage,Letter}}$*).*

By a simple query rewriting technique, we can obtain the mapping formula as follows:

$\mathcal{R}_{ut}$:*Grade(stuno, mark*$\downarrow_{r_{Percentage,Letter}}$*)*$\leftarrow_{int}\mathcal{R}_{uw}$:*Mark(stuno, mark*$\downarrow_{r_{Percentage,Letter}}$*).*

□

- **Dealing with the Entity Representation Predicates**. In the correspondence statement, we also introduce the representation predicate like *Represent([StuNo],x)* to indicate that the entity *x* in one schema is represented by the tuple of atomic element *[StuNo]* in another schema. It is possible that the same entity is represented by different tuples of atomic elements in different schemas. In this case, we can either ignore the representation predicate or introduce a new equivalent predicate *Equiv([a tuple],[another tuple])* during the query rewriting process to make the semantics explicit, as shown in the following example.

Example 14. Suppose the Course entity is represented by Course Number in $\mathcal{R}_{ut}$, while it is represented by Course Name in $\mathcal{R}_{uw}$,

$\mathcal{R}_{ut}$:*Course(CourseNo, CourseName, Description)* $\leftarrow_{int}$
$\mathcal{O}$:*COURSE(x),* $\mathcal{O}$:*hasNo(x,CourseNo),* $\mathcal{O}$:*hasName(x, CourseName),* $\mathcal{O}$:*hasDescription(x, Description), Represent([CourseNo],x).*

$\mathcal{R}_{uw}$:*Course(CourseNo, CourseName, Description)* $\leftarrow_{int}$
$\mathcal{O}$:*COURSE(x),* $\mathcal{O}$:*hasNo(x,CourseNo),* $\mathcal{O}$:*hasName(x, CourseName),* $\mathcal{O}$:*hasDescription(x, Description), Represent([CourseName],x).*

By introducing the equivalent predicate, *Equiv([CourseName],[CourseNo])*, to the second formula, the mapping can be obtained by rewriting as follows:

$\mathcal{R}_{ut}$:*Course(CourseNo, CourseName, Description)*$\leftarrow_{int}$
$\mathcal{R}_{uw}$:*Course(CourseNo, CourseName, Description), Equiv([CourseName], [CourseNo]).*

□

Apparently, the treatments of the specific domain value predicates and the entity representation predicates above do not change the setting for applying the mapping composition techniques. Given two schemas $\mathcal{R}_1$ and $\mathcal{R}_2$ and a domain ontology $\mathcal{O}$, and two finite sets of correspondence statements $\mathcal{R}_1 \leftarrow_{int(ext)} \mathcal{O}$ and $\mathcal{R}_2 \leftarrow_{int(ext)} \mathcal{O}$, the following corollary follows from the results in [MH03].

Corollary 1. *(Semantic Mapping)* Computing the semantic mapping $\mathcal{R}_1 \leftarrow_{int(ext)} \mathcal{R}_2$ is decidable and the resulting mapping contains a set of correspondence statements that captures the semantics specified in terms of the correspondences to the domain ontology. That is, for any query Q over $\mathcal{R}_1$w.r.t. to any database $D$of $\mathcal{R}_2$, the certain answers computed through $\mathcal{R}_1 \leftarrow_{int(ext)} \mathcal{R}_2$ coincide with the certain answers computed through two separate correspondences mediated by the domain ontology. □

Example 15. *(Continue our common scenario example.)* Example 11 gives a composition, $\mathcal{R}_{ut} \leftarrow_{int(ext)} \mathcal{O}$, of $\mathcal{R}_{ut} \leftarrow_{int(ext)} \mathcal{E}_{ut}$ and $\mathcal{E}_{ut} \leftarrow_{int(ext)} \mathcal{O}$. Example 8 presents several correspondence statements in the correspondences $\mathcal{R}_{uw} \leftarrow_{int(ext)} \mathcal{E}_{uw}$ and $\mathcal{E}_{uw} \leftarrow_{int(ext)} \mathcal{O}$. We compose $\mathcal{R}_{uw} \leftarrow_{int(ext)} \mathcal{E}_{uw}$ and $\mathcal{E}_{uw} \leftarrow_{int(ext)} \mathcal{O}$ to obtain $\mathcal{R}_{uw} \leftarrow_{int(ext)} \mathcal{O}$. The following formula is one of the set of semantic mapping formulas computed by the algorithm mentioned above.

$\mathcal{R}_{ut}$:Student(StuNo, Name, Address, "Grad", SupBy), $\mathcal{R}_{ut}$:Registration(StuNo, Year, Degree, Dept) $\leftarrow_{int}$ *$\mathcal{R}_{uw}$:GradStudent(StuNo, Name, Address, Year, Degree, SupBy), $\mathcal{R}_{uw}$:StudyIn(StuNo, Dept).*

It shows the left-hand side query expression over $\mathcal{R}_{ut}$ intensionally maps to the right-hand side query expression over $\mathcal{R}_{uw}$. There is not extensional mapping between them .

□

4.3 Deriving Semantic Mapping between Ontology-Mediated XML Schemas

Today, most commercial and scientific applications have facilitates for automatically exporting their data into XML form. Although the semantic web has proposed new data description language to assign semantics to data, it still needs to exchange data between XML structures because they are likely to exist for a long time. While the hierarchical structure of XML implicitly encodes logical relationships among elements, it does not assign semantic meaning to any particular elements. Interoperability is usually achieved in the real world by writing ad hoc translators between XML files. In this section, we will show that the correspondence continuum and the semantic encapsulation for XML schemas enable the automatic generation of semantic mappings between XML schemas.

We describe the main ideas of the algorithm and illustrate it via example using schemas in the two universities, *UofT and UofW*, scenario. We assume that each XML file comes with a schema, expressed in XML schema, which defines the terminology and the structural constraints for the XML file. In Example 10 we use tree-path following by variable name in square brackets to indicate the position of the variable in the XML hierarchical structure, and the following statement specifies the correspondence between the XML schema $\mathcal{X}_{ut}$ and the relational schema $\mathcal{R}_{ut}$ in the framework of the correspondence language:

$\mathcal{X}_{ut}$*:(s.Student.Level[$level], s.Student.StuNo[$StuNo], s.Student.Name[$Name], s.Student.Address[$Address],*
r.Registration.Degree[$Degree], r.Registration.Department[$Department],
*r.Registration.Year[$Year] | s*$\in$*UTDB.Students, r*$\in$*s.Registrations)* $\leftarrow_{ext}/\leftarrow_{int}$
$\mathcal{R}_{ut}$*:Student(StuNo, Name, Address, Level),* $\mathcal{R}_{ut}$*:Registration(StuNo, Year, Degree, Department).*

The following example shows the procedure for deriving semantic mapping between $\mathcal{X}_{ut}$ and $\mathcal{X}_{uw}$.

Example 16. The principle of correspondence continuum and semantic encapsulation guarantees the explicit presentation of the semantics of XML schemas in terms of correspondences. Given the correspondences of $\mathcal{X}_{ut} \leftarrow_{int(ext)} \mathcal{R}_{ut}$, $\mathcal{R}_{ut} \leftarrow_{int(ext)} \mathcal{E}_{ut}$, and $\mathcal{E}_{ut} \leftarrow_{int(ext)} \mathcal{O}$ as well as the correspondences of $\mathcal{X}_{uw} \leftarrow_{int(ext)} \mathcal{R}_{uw}$, $\mathcal{R}_{uw} \leftarrow_{int(ext)} \mathcal{E}_{uw}$, and $\mathcal{E}_{uw} \leftarrow_{int(ext)} \mathcal{O}$, the semantic mapping $\mathcal{X}_{ut} \leftarrow_{int(ext)} \mathcal{X}_{uw}$ is derived in following steps:

step 1: Correspondence Composition. The first step is to compose the compatible correspondences to obtain the direct correspondence from XML schemas to the domain ontology. After composition, we obtain the following direct correspondences.

$\mathcal{X}_{ut}$*:(s.Student.Level["Grad"], s.Student.StuNo[$StuNo], s.Student.Name[$Name], s.Student.Address[$Address],*
r.Registration.Degree[$Degree], r.Registration.Department[$Department],
*r.Registration.Year[$Year] | s*$\in$*UTDB.Students, r*$\in$*s.Registrations)*
$\leftarrow_{int}$
$\mathcal{O}$ *:GRAD(x),* $\mathcal{O}$ *:REGISTRATION(y),* $\mathcal{O}$ *:registerAt(x,y),* $\mathcal{O}$ *:hasStuno(x, StuNo),* $\mathcal{O}$ *:hasName(x, z),* $\mathcal{O}$ *:PERSON-NAME(z),* $\mathcal{O}$ *:hasFirstName(z, y1),* $\mathcal{O}$ *:hasLastName(z,y2), concat(y1,y2,Name),* $\mathcal{O}$ *:hasAddress(x, Address),* $\mathcal{O}$ *:inYear(y, Year),* $\mathcal{O}$ *:withDegree(y, Degree),* $\mathcal{O}$ *:ofDept(y, Department).*

$\mathcal{X}_{uw}$*:(s.UndergradStudent.sno[$sno], s.UndergradStudent.Name[$name], s.UndergradStudent.Address[$address],*
s.UndergradStudent.Year[$year], s.UndergradStudent.Department[$department],
s.UndergradStudent.Degree[$degree], g.GradStudent.sno[$gSno], g.GradStudent.Name[$gName],
g.GradStudent.Address[$gAddress], g.GradStudent.Year[$gYear], g.GradStudent.Department[$gDepartment],
*g.GradStudent.Degree[$gDegree] | s*$\in$*UWDB.UndergradStudents, g*$\in$*UWDB.GradStudents)*
$\leftarrow_{int}$
$\mathcal{O}$ *:UNDERGRAD(x),* $\mathcal{O}$ *:REGISTRATION(y),* $\mathcal{O}$ *:registerAt(x,y),* $\mathcal{O}$ *:hasStuno(x, sno),* $\mathcal{O}$ *:hasName(x, z),*
$\mathcal{O}$ *:PERSON-NAME(z),* $\mathcal{O}$ *:hasFirstName(z, y1),* $\mathcal{O}$ *:hasLastName(z,y2), concat(y1,y2,name),*
$\mathcal{O}$ *:hasAddress(x, address),* $\mathcal{O}$ *:inYear(y, year),* $\mathcal{O}$ *:withDegree(y, degree),* $\mathcal{O}$ *:ofDept(y, department),*
$\mathcal{O}$ *:GRAD(u),* $\mathcal{O}$ *:REGISTRATION(v),* $\mathcal{O}$ *:registerAt(u,v),* $\mathcal{O}$ *:hasStuno(u, gSno),* $\mathcal{O}$ *:hasName(u, w),*
$\mathcal{O}$ *:PERSON-NAME(w),* $\mathcal{O}$ *:hasFirstName(w, v1),* $\mathcal{O}$ *:hasLastName(w,v2), concat(v1,v2,gName),*
$\mathcal{O}$ *:hasAddress(u, gAddress),* $\mathcal{O}$ *:inYear(v, gYear),* $\mathcal{O}$ *:withDegree(v, gDegree),* $\mathcal{O}$ *:ofDept(v, gDepartment).*

step 2: Query Rewriting. The second step is to obtain the maximally-contained rewriting of the right-hand side of the first correspondence formula in terms of the view which has body as the right-hand side of the second correspondence formula. After unfolding the view in the rewriting, we proceed to obtain the maximally-contained rewriting of the unfolding in terms of the left-hand side of the second correspondence formula. By appropriate variable renaming, the following formula is obtained through rewriting.

$\mathcal{X}_{ut}$*:(s.Student.Level["Grad"], s.Student.StuNo[$gSno],s.Student.Name[$gName], s.Student.Address[$gAddress], r.Registration.Degree[$gDegree], r.Registration.Department[$gDepartment], r.Registration.Year[$gYear] | s∈UTDB.Students, r∈s.Registrations)*

$\leftarrow_{int}$

$\mathcal{X}_{uw}$*:(s.UndergradStudent.sno[$sno], s.UndergradStudent.Name[$name], s.UndergradStudent.Address[$address], s.UndergradStudent.Year[$year], s.UndergradStudent.Department[$department], s.UndergradStudent.Degree[$degree], g.GradStudent.sno[$gSno], g.GradStudent.Name[$gName], g.GradStudent.Address[$gAddress], g.GradStudent.Year[$gYear], g.GradStudent.Department[$gDepartment], g.GradStudent.Degree[$gDegree] | s∈UWDB.UndergradStudents, g∈UWDB.GradStudents).*

step 3: Tree Pruning. Finally, prune the right-hand side XML schema tree by detecting those variables which are not been mapped in the above mapping formula. The corresponding tree paths where the unmapped variables are located can be eliminated from the mapping. Figure 13 shows the final mapping between $\mathcal{X}_{ut}$ and $\mathcal{X}_{uw}$ graphically.

□

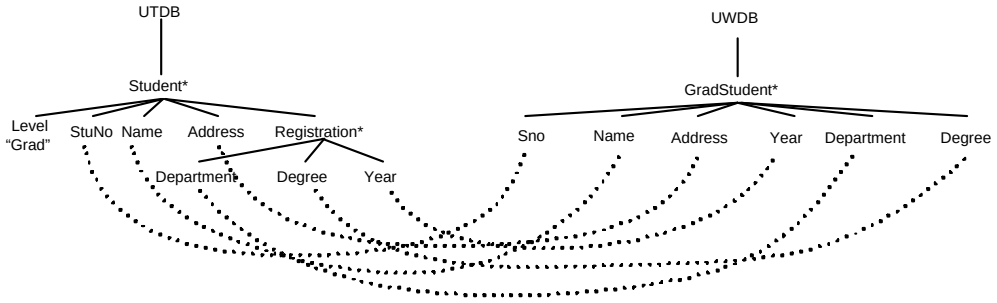

Figure 13: Semantic Mapping between $\mathcal{X}_{ut}$ and $\mathcal{X}_{uw}$.

As shown in the above example, analogous to the flat relational schemas, XML schemas can also take advantage of the semantic encapsulation in terms of the correspondences to connect them together by semantic mappings for information exchanging. Our XML schema notation is inspired by the nested relational model in [PVM$^+$02]. The addition of variables in the square brackets following tree paths makes the notation suitable for our correspondence language. In doing this, we have reconciled XML schema, relational schema, Entity-relational schema, and domain ontology described in Description Logics in a single correspondence continuum. The correspondence continuum is such a fertile land of semantics that interoperability will be easily attained for a wide variety of data sources.

5 Related Work

The goal of this paper is to propose a general framework for capturing data semantics appearing in an open-ended environment in terms of correspondence. The study of semantics has been scattered

in diverse communities, and it is impossible to compile an exhaustive list for exposing all efforts and results. In this section, we will expand our scope of references to cover as many representative works in database and knowledge representation communities as possible. The following exposition starts from the efforts in data integration, manifested in database schema manipulation and real application systems, goes through works in knowledge representation and the semantic web, and concludes with recent mapping management efforts discussed in various fields.

Problem of schema integration and merging emerged from earlier database design context such as view integration from a set of external views. Later, data gathering and sharing in federated databases systems call for a global integrated schema in the same spirit. The key is to explore the semantics of a set of schemas to discover similarity. [BLN86] makes a comprehensive survey and analysis about various causes of schema diversity. [Kos91] studies schema merging in terms of a formal mathematic schema model; [SP94] uses the real world states as the semantics of the schema elements; and [KLK91] proposes the schematic discrepancy problem. These works pioneered the study in heterogeneity reconciliation.

The efforts to discover semantic relations between elements in different schemas result in various schema matching techniques. [RB01] classifies matchers into schema- and instance-level, element- and structure-level, language- and constraint-level, and individual- and combination methods. In particular, [MBR01], [MGMR02], and [NM01] approach the schema matching problem by employing linguistic techniques as well as by exploring syntactic structures. [BSZ03] studies a special type of schema, namely, the concept hierarchy, and shifts the problem from computing linguistic and structural similarities to deducing relations between sets of logical formulas. The WordNet [MFT+03] serves the background theory for deduction. Moving one step further, schema mapping [PVM+02, MHH00] discovers the formal queries over input schemas in accordance with the value correspondences specified by users or matchers. Data dependence theory and the traditional chase technique are the core components of the query discovery process. [FKMP03] studies the general data exchange problem and gives a theoretical justification for the results in [PVM+02, MHH00]. Semantic mapping between heterogeneous data sources started to become a significant interest in database community.

Specifying semantic relation between schemas plays a central role in data integration systems [Len02] in which a set of sources links to a virtual global schema and the materialized data in source provides answers to global queries. Especially, the local-as-view (LAV) formalism results in the extensive study in view-based query answering and rewriting [Hal01, Hal00, LMSS95, Ull00]. The complexity[AD98] varies according to the expressiveness of the view definition languages and the query language. There was a flurry of study in data integration application systems in mid 90's. The salient ones include Carnot project [CHS91], InfoSleuth project [BBB+97], SIMS project [AKS96], Information Manifold project [LSK96], and OBSERVER project [MKSI96]. The commonality of these systems is that many have realized that ontology as the agreed upon specification of a domain with richer semantics can serve the mediating role for integration.

Different from the data integration structure, peer-to-peer data management system (PDMS) addresses query answering problem over a set of more loosely coupled data sources. [BGK+02] describes local relational models as a formalism for mediating between different peers in a PDMS; [HIMT03] and [HIST03] provide mapping languages between peers and give algorithms for answering queries over peer-to-peer infrastructure by leveraging well-understood database query languages. Realizing mapping is the core component for a variety of model management problems, the proposal of generic model management [BHP00, Ber03] treats both mapping and model as

first-class citizens which can be manipulated by model-at-a-time and mapping-at-a-time operators. However, both the PDMS and the the generic model management leave the mapping generation problem untouched.

Representing human knowledge in terms of symbolic structures is a long-standing problem. Description Logics [BN03] developed in knowledge representation community finds its way into data management tasks. [BLR03] and [Bor95] show that DLs are useful not only for specifying schemas, but also for obtaining descriptive or intensional answers. Recently, the distributed Description Logics [BS03] extends the DLs formalism with the ability to handle complex mappings between domains, through the use of bridge rules. In addition, DLs also has served successfully as ontology languages. Ontology integration and the use of ontology in assisting semantic coordination have their own significant interests and have been studied in [CGL01, C$^+$01, CLN98, CLN99, CCGL02]. Introducing the notion of context, [KS96b, KS96a] explore the approaches based on the capture and representation of meta-data, contexts, and ontologies to manage semantic heterogeneity within the framework of DLs.

Data on the Web raise more challenges to interoperability. The semantic web [BLHL01] is an extension of the current web in which information is given well-defined semantics, better enabling computers and people to work in cooperation. New data model such as RDF [KC03] and new markup language such as OWL [DS03] have been developed inspired by technology from knowledge representation. Interoperability would be easily attained as long as the agreed upon web ontologies have been used and shared among information providers [HBLM02]. However, it is widely accepted that there will be many heterogeneous ontologies existing on the semantic web and traditional data model such as XML will persist. Addressing this, [B$^+$03] studies contextualized ontologies and the semantic connections between them, and [LS03] discusses the interoperability on XML. Our work on correspondence continuum extends the one-tier mapping structure to chains of semantic correspondences. Interoperability on various data models including XML is attained by the explicit representation of data semantics.

With regards to mapping management, a variety of mapping formalisms ranging from simple value correspondence to formal logical expressions have been proposed in literature. [CL93] proposes a logic approach to the problem of both expressing inter-schema knowledge and reasoning about it. Its salient feature is the distinction of intensional and extensional mapping representations for capturing different cognitive meanings. Mappings in [C$^+$01] are augmented with domain value relations in terms of conversion, matching, and merging. [MB$^+$02] discusses a very high-level framework for defining representation of mappings with associated semantics, and [MH03] studies the mapping composition problem and gives a composition algorithm for a specific class of query language. Finally, [FS03] presents a theoretical ground and principled methodology for mapping based on channel theory, a mathematical theory of semantic information flow. We expect that the similar application of category theory will provide a solid theoretical foundation for our semantic correspondence work in the future.

6 Conclusions and Future Work

The heterogeneity of data sources in terms of various structures such that XML formats, relational schemas, object-oriented schemas, and entity-relational schemas and their autonomy as well as distributed locations are the major burdens for interoperability over the Internet infrastructure. The

years investigation of data integration systems in database community relies on a globally consolidated schema and mappings between sources and the global schema to enable query answering without paying attention to data source locations. The semantics of data sources as well as the global schema are entrusted to a small set of users or applications. However, the Web does not provide the chance for any groups of people to have knowledge about all data sources participating in information sharing. Consequently, the semantic web initiative proposes an approach for presenting data semantics explicitly using ontology, and it certainly will lead significant changes in the study of data integration over the Web. Although the vision of the semantic web is compelling, disconnectivity emerges. On the one side, traditional relational databases and recent XML files have been used widely for many applications and mature techniques for database design, query language, and optimization have been well developed in database community. Furthermore, the XML format has been accepted as the standard structure to export data on the Web for information exchanging. It is no doubt that data in these formats will persist indefinitely and continue to grow. On the other side, the semantic web requires a massive use of ideas and techniques from knowledge representation. It requires everything to be built up from scratch. New data models and languages such as RDF/S, DAML+OIL, and OWL as well as standardized ontologies will play a major role, and semantic data are presented explicitly to users and software agents. The lesson we learned from both sides is that capturing data semantics and sticking on respective approaches is the key to interoperability.

In this paper, we have presented our two principles for capturing data semantics: the correspondence continuum and the semantic encapsulation. Representing meaning in terms of correspondence is not new, and the model-theoretic semantics of standard logic can be seen as a correspondence theory of meaning. However, the practices in information systems modeling and databases design show that the two-tier model-theoretic analysis is inadequate for analyzing data semantics. By adopting the correspondence continuum and proposing the semantic encapsulation, we have made the following specific contributions. First, we described the correspondence continuum as a semantic soup for locating a spectrum of semantic relations among given intentional structures - language, process, schema, and models. The continuum is grounded on domain ontology. Focusing on three common types of database structures - XML format, relational schema, and entity-relational model, we proceeded to describe the tight correspondences between them in the designing phase. Second, we proposed the principle of semantic encapsulation. Each schema comes with its semantics in terms of the correspondence to other structures. By leveraging database query languages, we proposed a formal correspondence language which is capable of capturing both intensional and extensional correspondence between structures. Third, we described two reasoning tasks taking advantage of correspondences. Leveraging a rich body of techniques for view-based query answering and data integration formalisms, we showed that correspondences can be composed along the lineage line upwards until reaching the domain ontology with the richest semantics. Further, we demonstrated that algorithm for generating semantic mappings between arbitrary schemas exists indeed. Finally, we gave a procedure for deriving semantic mappings between XML schemas. The novel XML notation combining with the correspondence language achieves the goal of XML interoperability on the Web.

There are many lines of future research and work ahead. First, we are currently pursuing the implementation of our framework in terms of a Web-accessible tool which includes a domain ontology repository, a conceptual schema repository, and a logical schema repository. The tool allows user to browse the repositories and create new schemas or set correspondences between schemas. Generating semantic mappings between schemas will be provided as web services. Second, the reasoning

tasks are based on our two assumptions: single line of correspondences and single domain ontology. In reality, a schema structure can correspond to any other schema structures, and different versions of domain ontologies exist indeed. We will proceed to develop techniques for such a complicated situation by combining techniques from ontology articulation. Third, we notice that the generic model management [Ber03] treats models and mappings as abstractions that can be manipulated by model-as-a-time and mapping-as-a-time operators. Despite mapping is the significant interest and all the operators rely on it, it is by no means an easy task to generate mappings between models automatically without explicit semantic presentation. We are interested in placing our semantic encapsulation principle into the generic model management framework. Fourth, as we mentioned before, to make the semantic web succeed, traditional databases and XML files have to be covered. Our correspondence approach provides a promising way to reconcile the two worlds. Therefore, new web correspondence language needs to be developed in addition to web ontology language. Finally, we are interested in pursuing theoretical and philosophical issues regarding correspondence and mapping more deeply. We have seen that the standard logic is inadequate in dealing with intensional correspondence. Although we have distinguished the logical symbols for intensional and extensional correspondences in our correspondence language, we expect to bring in more results of the theory of 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.

[BGK+02] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Database Management for Peer-to-Peer Computing: A Vision. In *The International Workshop on the Web and Databases*, 2002.

[BHP00] Phillip A. Bernstein, Alon Y. Halevy, and Rachel A. Pottinger. A vision for management of complex models. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):55–63, 2000.

[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. D. Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *the 10th Italian Conf. on Database Systems*, pages 161–168, 2002.

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

[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.

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

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. In *ICDT'03 (LNCS 2572)*, pages 207–224, 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.

[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.

[KC03] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Working Draft 23, http://www.w3.org/TR/rdf-concepts, January 2003.

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

[Kos91] Anthony S. Kosky. Modeling and Merging Database Schemas. Technical report, University of Pennsylvania, 1991.

[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.

[LS03] Laks V. S. Lakshmanan and Fereidoon Sadri. Interoperability on XML. In *ISWC'03*, 2003.

[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.

[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] R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema Mapping as Query Discovery. Technical Report CSRG-412, University of Toronto, 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.

[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.

[PSS02] P. Patel-Schneider and J. Simeon. Building the semantic web on XML. In *ISWC'02*, 2002.

[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.

[Rei84] Raymond Reiter. Towards a logical reconstruction of relational database theory. In *M. Brodie, J. Mylopoulos, J. Schmidt eds: On Conceptual Modelling, springer-verlag*, pages 191–233, 1984.

[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.

[Ull00] Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.