Semantic and Schematic Similarities between Database Objects: A Context-based approach

Vipul Kashyap
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Amit Sheth
LSDIS, Department of Computer Science
University of Georgia
415 GSRC, GA 30602-7404

September 25, 1995

Abstract

In a multidatabase system, schematic conflicts between two objects are usually of interest only when the objects have some semantic similarity. We use the concept of *semantic proximity*, which is essentially an *abstraction/mapping* between the domains of the two objects associated with the *context of comparison*. An explicit though partial context representation is proposed and the specificity relationship between contexts is defined. The contexts are organized as a meet semi-lattice and associated operations like the greatest lower bound (*glb*) are defined. The context of comparison and the type of abstractions used to relate the two objects form the basis of a semantic taxonomy. At the *semantic level*, the intensional description of database objects provided by the context is expressed in a description logic language. *Schema correspondences* are used to store mappings from the semantic level to the data level and are associated with the respective contexts. Inferences about database content at the federation level are modeled as changes in the context and the associated schema correspondences. We try to reconcile the dual (schematic and semantic) perspecitves by: enumerating *possible semantic similarities* between objects having schema and data conflicts, and modeling schema correspondences as the projection of semantic proximity *wrt* context.

1 Introduction

Many organizations face the challenge of interoperating among multiple independently developed database systems to perform critical functions. With high interconnectivity and access to many information sources, the primary issue in the future will not be how to efficiently process the data that is known to be relevant, but to determine which data is relevant [She91]. Thus, the fundamental question in interoperability is that of identifying objects in different databases that are semantically related, and then resolving the schematic differences among semantically related objects. In this paper, we are interested in the reconciliation of the semantic and schematic perspectives and use it as a step towards *information focusing* and *correlation* across multiple databases.

We characterize the degree of semantic similarity between a pair of objects using the concept of **semantic proximity** [SK92]. It is based on the premise that it is essential to associate the **abstractions/mappings** between the objects with the **context of comparison** for capturing the semantic similarity between them. Other researchers in the field of multidatabases have also made observations that are similar in principle, but different in details [ON93, SSR92, YSDK91]. This association of context with abstractions represents the first step in achieving the reconciliation between the semantic and schematic perspectives.

To appear in *The VLDB Journal, The International Journal on Very Large Data Bases*, P. Apers, H. Schek, J. Gray and S. Su (editors-in-chief)

Inadequacies of purely structural and mapping based methods are discussed and explicit representation of context is proposed to resolve some inadequacies. Computational benefits of representing context are also discussed. We propose a partial representation of context as a collection of contextual coordinates and their values. This representation is used to describe objects and the constraints which they must satisfy in an intensional manner. The meaning of the contextual coordinates and their values are explained by expressing the context in a description logic like language [BS85, BBMR89].

In order for a context representation to be useful for semantic interoperability in multidatabases, it is important to have automatic ways of comparing and manipulating them. Based on the proposed representation of context, we define the specificity relationship between two contexts. A definition of the specificity relationship and the *glb* and other operations on contexts are presented. The specificity relationship induces a partial order such that for any two contexts, there exists a *glb* leading to the organization of the context set as a meet semi-lattice.

The semantic proximity descriptor consists of context and abstraction as its main components. Depending on the values assumed by these two components, we define a data model independent taxonomy of semantic similarities. The possible values of the first component can be contexts constructed using the various operations mentioned above. Classification or taxonomies of *schematic differences* appear in Multidatabase literature. However, purely schematic considerations do not suffice to determine the similarity between objects [FKN91][SG89]. We try to reconcile the two perspectives by enumerating the possible semantic similarities between objects having schematic and data conflicts.

Even though the representation of semantics better enables us to represent the similarities between the various objects, we also need to be able to capture structural similarities in a mathematical formalism for reasoning on the computer. We define the concept of **schema correspondences** to capture the structural similarities between the objects. They are also associated with the context in which the semantic proximity is defined. We reconcile the semantic and schematic perspectives by defining the schema correspondence as a projection of the semantic proximity *wrt* context. The semantics of the projection operation are captured in the rules of the algebra enumerated in Appendix A.1.

The overall organization of the paper is as follows. In Section 2 we present a model to represent semantic similarities among objects. In Section 3 we discuss the rationale for representation of context in a multidatabase environment and propose an explicit, though partial, representation of context. The associated operations for reasoning about and manipulating the context representations are also defined. In Section 4 a taxonomy of the various types of possible semantic similarities between the various objects is presented. In Section 5 we discuss a broad class of schematic differences and the possible semantic similarities between objects having those differences. In Section 6 we define a uniform formalism for representation of structural similarity. It is associated with the context and is defined as the projection of semantic similarity. Examples illustrating the operations from an algebra describing the projection operation (Appendix A.1) are presented. A discussion of related work is presented in Section 7. Conclusions and future work are presented in Section 8.

2 Semantic Similarities between Objects

In this section, we discuss the concept of *semantic proximity* which characterizes **semantic** similarities between objects. We distinguish between the *real world*, and the *model world* which is a representation of the real world. Like the work in semantic data modeling [HK87, PM88], we endeavor to capture some of the important semantic information about the real world and represent it in the model world. However, over and above the semantics of the data we also

attempt to capture semantics of queries and applications. This enables us to support semantics-based information focusing and correlation across multiple databases *wrt* an application.

Attempts have been made to capture the similarity of objects by using mathematical tools like value mappings between domains and abstractions like generalization, aggregation, etc. However, it is our belief that the *real world semantics* (RWS) of an object[1] cannot be captured sufficiently using mathematical formalisms. The term "object" in this paper refers to an object in the model world (i.e., a representation or intensional definition in the model world, e.g., an object class definition in object-oriented models or relation in relational models) as opposed to an entity or a concept in the real world. These objects may model information at any level of representation, such as the *attribute* or *entity* level.

We need to understand and represent more knowledge to capture the semantics of relationships between objects. This knowledge should be able to capture the **context** of comparison of the objects and the **abstraction** relating the domains of the two objects. Attempts to partially represent such extra knowledge includes the use of meta-attributes [SSR92] and building and partitioning ontologies into micro-theories [Guh90].

Attempts to represent context and abstraction as suggested above have been reflected in the techniques and representational constructs used by various practitioners and researchers in the field of multidatabases. The model for semantic proximity defined in this section has been influenced by these attempts. Some significant attempts are the **semantic proximity** proposal by Sheth and Kashyap [SK92], the **context building** approach by Ouksel and Naiman [ON93], the **context interchange** approach by Sciore et al. [SSR92] and the **common concepts** approach by Yu et al. [YSDK91]. We relate the above attempts to semantic proximity. A detailed discussion of these can be found in [KS95c].

Semantic Proximity: A model for Semantic Similarity

Given two objects O_1 and O_2, the *semantic proximity* between them is defined by the 4-tuple given by [SK92]:

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

where D_i is domain of O_i and S_i is state of O_i.

- The first component denotes the context in which the two objects O_1 and O_2 are being compared. This context may be the same, different, or related in some manner to the context(s) in which the objects O_1 and O_2 are defined.
- The second component identifies the abstraction/mapping used to relate the domains of the objects, O_1 and O_2.
- The third component enumerates the domain definitions of the objects, O_1 and O_2. The domains may be defined by either enumerating the values as a set or by using existing type definitions in the database.
- The fourth component enumerates the states of the objects, which are the extensions of the objects recorded in their respective databases at a particular time.

In Figure 1 we have illustrated the definition of the semantic proximity between two objects O_1 and O_2 in the database. Context(O_1) and context(O_2) represent the contexts (referred to as *definition contexts* later in the paper) in which the objects O_1 and O_2 are mapped from the real world to the model world. Context(O_1,O_2) refers to the context in which the objects are being compared.

[1] The term "real world semantics" distinguishes from the "(model) semantics" that can be captured using the abstractions in a semantic data model. Our definition is also intensional in nature, and differs from the extensional definition of Elmasri et al. [ELN86] who define RWS of an object to be the set of real world objects it represents.

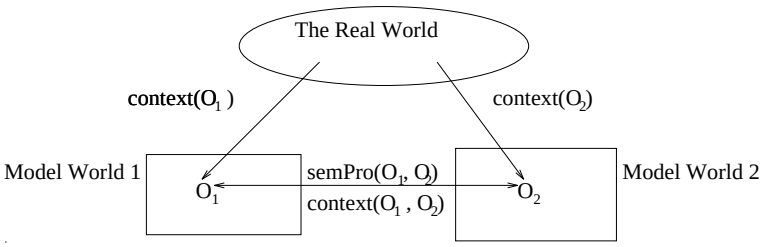

Figure 1: Semantic Proximity between two Objects

Context: The semantic component

The context is the key component in capturing the semantics related to an object's definition and its relationships to other objects. Alternatives discussed in multidatabase literature for representing context are as follows.

- In [ON93], context is defined as the knowledge that is needed to reason about another system, for the purpose of answering a query. It is specified as a set of assertions identifying the correspondences between various schema elements.
- In [SSR92], context is defined as the meaning, content, organization and properties of data. It is modeled using metadata associated with the data.
- In [YSDK91], *common concepts* are proposed to characterize similarities between attributes in multiple databases.
- When using a well defined ontology, such as Cyc [Guh90], a well defined partition (called *Microtheory*) of the ontology is assigned a context.
- A context may be identified or represented using the following [SK92].
  - By association with a database or a group of databases.
  - As the *relationship* in which an entity participates.
  - From a schema architecture (e.g., the multidatabase or federated schema architecture of [SL90]), a context can be specified in terms of an *export schema* (a context that is closer to the database) or an *external schema* (a context that is closer to the application).
  - At a very elementary level, as a *named collection* of domains of objects.

A context may be used in several ways to capture the relevant semantics. A context may be associated with an object to specify the assumptions used in its design and its relationships with other objects. However, the term context in semPro refers to the context in which a particular semantic similarity holds between two objects. As we shall see later, the context in SemPro need not be the exactly the same as the contexts associated with the objects.

Abstractions/Mappings: The structural component

We use the term abstraction to refer to the relation between the domains of the two objects. Mapping between the domains of objects is the mathematical expression to denote the abstractions. However, since abstractions by themselves cannot capture semantic similarity, they have to be associated either with the context [KS93] or with extra knowledge in order to capture the RWS. Some of the proposals are as follows.

- In [SK92], abstractions are defined in terms of value mappings between the domains of objects and are associated with the context as a part of the semantic proximity.
- In [ON93], mappings are defined between schema elements called *inter schema correspondence assertions* or ISCAs. A set of ISCAs under consideration define the context for integration of the schemas.

- In [SSR92], mappings called *conversion functions* are associated with the meta-attributes which define the context.
- In [YSDK91], the attributes are associated with "common concepts". Thus the mappings (relationship) between the attributes are determined through the extra knowledge associated with the concepts.

Some useful and well-defined abstractions are:

Total 1-1 value mapping For every value in the domain of one object, there exists a value in the domain of the other object and vice versa.

Partial many-one mapping In this case some values in the domain of one of the objects might remain unmapped, or a value in one domain might be associated with many values in another domain.

Generalization/Specialization One domain can generalize/specialize the other, or domains of both the objects can be generalized/specialized to a third domain.

Aggregation One domain can be an aggregation or a collection of other domains.

Functional Dependencies The values of one domain might depend functionally on the other domain.

ANY This is used to denote that any abstraction such as the ones defined above may be used to define a mapping between the domains of two objects.

NONE This is used to denote that there is no mapping defined between the domains of two objects.

Domains of the Objects

Domains refer to the sets of values from which the objects can take their values. When using an object-oriented model, the domains of objects can be thought of as types, whereas the collections of objects might themselves be thought of as classes. A domain can be either **atomic** (i.e., cannot be decomposed any further) or composed of other atomic or composite domains. The domain of an object can be thought of as a subset of the cross-product of the domains of the properties of the object (Figure 2). Analogously, we can have other combinations of domains, such as union and intersection of domains.

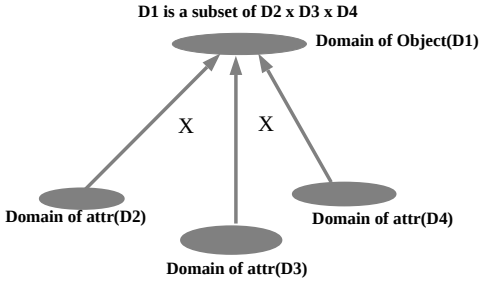

Figure 2: Domain of an Object and its Attributes

An important distinction between a context and a domain should be noted. One of the ways to specify a context is as a named collection of the domains of objects, i.e. it is associated with a group of objects. A domain on the other hand is a property of an object and is associated with the description of that object.

States (extensions) of the Objects

The state of an object can be thought of as an extension of an object recorded in a database or databases. However, this extension must not be confused with the actual state of the entity being modeled according to the Real World Semantics. Two objects having different extensions can have the same state Real World Semantics (and hence be semantically equivalent).

3 Explicit context representation in a multidatabase environment

In this section we discuss the inadequacies of purely structural and mapping based methods to represent object similarity and how representing context in the model world helps solve some of them. We also discuss computational advantages of representing context in the model world and propose an appropriate representation of context as a collection of contextual coordinates and their values. The contextual coordinates and their values may be chosen from a previously defined ontology of concepts.

We view ontology as the symbolic layer closest to concepts in the real world. An ontology may be defined as the specification of a representational vocabulary for a shared domain of discourse which may include definitions of classes, relations, functions and other objects [Gru93]. Criteria for constructing contexts from an ontology are discussed in [KS95a].

We shall discuss a partial representation of context and equivalent expressions in a description logic language. We shall also define operations for automatic ways of comparing (e.g., deciding whether one context is more general than the other) and manipulating contexts (e.g., taking the greatest lower bound of two contexts). A brief discussion of issues relating to the language for representing contexts and the ontologies from which the contexts may be constructed are also discussed.

3.1 Rationale for Context representation

In characterizing the similarity between objects based on the semantics associated with them we have to consider the RWS of an object. It is not possible to completely define what an object denotes or means in the model world [SG89]. We propose the **context** of an object as the primary vehicle to capture the RWS of the object. The context in which two objects are being compared and the associated abstraction/mapping helps to capture the semantic aspect of the relationship between two objects (Figure 1). We argue for the need for representing context by showing the inadequacy of purely structural representations. We also discuss the computational benefits of representing context.

3.1.1 Inadequacy of purely Structural Representations

It has been suggested by Sheth and Gala/Kashyap [SG89][KS94b] and Fankhauser et al. [FKN91] that the ability to represent the structure of an object does not help capture the real world semantics of the object. It is not possible to provide a structural and hence a mathematical definition of the complex notion of real world semantics. In [LNE89], a one-to-one mapping is assumed between the attribute definition and the attribute's real world semantics. They define an attribute in terms of fixed descriptors such as *Uniqueness, Lower/Upper Bound, Domain, Scale* etc. which are used to generate mappings between two attributes. They are also used to determine the equivalence of attributes. However what they establish is the structural equivalence of these attributes which is necessary but not sufficient to determine the semantic equivalence of the attributes.

Consider two attributes *person-name* and *department-name*. We may be able to define a mapping between the domains of these two attributes, but we know that they are not semantically equivalent. In order to be able to capture this lack of equivalence, we propose the mappings between the domains of the attributes be made *wrt* a context. We define two objects to be semantically equivalent if it is possible to define mappings *wrt* all known and coherent contexts and the definition contexts should be coherent *wrt* each other. Definition contexts and the notion of coherence is defined later in this section. Since the definition contexts of *person-name* and *department-name* are not coherent (one identifies an animate and the other identifies an inanimate object), they are not defined to be equivalent.

3.1.2 Computational benefits of representing context

In [Sho91], Shoham has discussed the computational benefits that might accrue in modeling and representing context in AI and Knowledge-Based systems. We believe that there are similarities between AI/Knowledge-Based and multidatabase systems that suggest context representation in a multidatabase system for a clean and efficient handling of information.

Economy of representation: In a manner akin to database views, contexts can act as a *focusing mechanism* when accessing the component databases of a multidatabase system. They can be a *semantic summary* of the information in a database or group of databases and maybe able to capture semantic information which cannot be expressed in the data definition model of the databases. Thus unnecessary details can be abstracted from the user. Examples detailing this are enumerated in Section 6.2.

Economy of reasoning: Instead of reasoning with the information present in the database as a whole, reasoning can be performed with the context associated with a database or a group of databases. This approach has been used in [KS94a] for information resource discovery and query processing in Multidatabases.

Handing Inconsistent Information: In a multidatabase system, where databases are designed and developed independently, it is not uncommon to have information in one database inconsistent with information in another. As long as information is consistent within the context of the query of the user, inconsistency in information from different databases may be allowed. This is discussed in Section 5.3.

Flexible semantics: A big fallout of associating abstractions/mappings with the context in the semantic proximity model (Section 2) is that the same two objects can be related to each other differently in two different contexts. This is because two objects might be semantically closer to each other in one context as compared to the other.

3.2 A partial Context representation

There have been attempts to represent the similarity between two objects in databases. In [LNE89], a fixed set of descriptors define essential characteristics of the attribute and are used to generate mappings between them. We have discussed with the help of an example how they do not guarantee semantic similarity. Thus, any representation of context which can be described by a fixed set of descriptors is not appropriate.

The descriptors (or meta-attributes) are not fixed but are dynamically chosen to model the characteristics of the application domain in question. It is not possible a priori to determine all possible meta-attributes which would completely characterize the semantics of the application domain. This leads to a *partial* representation of context. We represent context as a collection of contextual coordinates (meta-attributes) as follows:

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

We shall explain the meaning of the symbols C_i and V_i by using examples and by enumerating the corresponding CLASSIC expressions. Table 1 shows how our context descriptions can be mapped to CLASSIC [BBMR89], a description logic language. Using CLASSIC, it is possible to define primitive classes and in addition specify classes using intensional descriptions phrased in terms of necessary and sufficient properties that must be satisfied by their instances. The intensional descriptions may be used to express the collection of constraints that make up a context. Also, each C_i roughly corresponds to a role and each V_i roughly corresponds to fillers for the role the object must have.

- C_i, $1 \leq i \leq k$, is a contextual coordinate denoting an aspect of context.

- C_i may model some characteristic of the subject domain and may be obtained from a domain specific ontology (discussed later in this section).
- C_i may model an implicit assumption in the design of a database.
- C_i may or may not be associated with an attribute A_j of an object O in the database.

Contextual coordinates and Values, C_{def}(O), C_q	**CLASSIC descriptions**
<(C_1, V_1) ... (C_k, V_k)>	(**AND** O (**ALL** C_1 V_1) ... (**ALL** C_k, V_k))
<(C_i, O_i∘ <(C_j, V_j)>)>	(**AND** O (**ALL** C_i (**AND** O_j (**ALL** C_j V_j))))
<(C_i, X) (C_j, X)>	(**AND** ANSWER (**SAME-AS** (**FILLS** C_i) (**FILLS** C_j)))
<(C_i, X∘ <(C_j, V_j)>)>	(**AND** ANSWER (**FILLS** C_i (**ALL** C_j V_j)))

Table 1: Contextual coordinate, value pairs and the corresponding CLASSIC expressions

The value V_i of a contextual coordinate C_i can be represented in the following manner:

- V_i can be a variable.
  - It can be unified (in the sense of Prolog) with another variable, a set of symbols, an object or type defined in the database or another variable.
  - It can be unified with another variable associated with a context.
  - It can be used as a place holder to elicit answers from the databases and impose constraints on them.

  Example:
  Suppose we are interested in people who are authors and who hold a post. We can represent the query context C_q (discussed later in this section) as follows:
  C_q = <(author, X) (designee, X)>
  The same thing can be expressed in a Description Logic (DL) as follows:
  C_q = (**AND** ANSWER (**SAME-AS** (**FILLS** author) (**FILLS** designee)))
- V_i can be a set.
  - The set may be an enumeration of symbols from a domain specific ontology.
  - The set may be defined as the extension of an object or as elements from the domain of a type defined in the database.
  - The set may be defined by posing constraints on pre-existing sets.

  Example:
  Suppose we want to represent the assumptions implicit in the design of the object EMPLOYEE in a database. We can represent this as the definition context of EMPLOYEE, C_{def}(EMPLOYEE) as follows:
  C_{def}(EMPLOYEE) = <(employer, [**Deptypes** ∪ {restypes}])(article, PUBLICATION)>
  The same thing can be expressed in a DL as follows:
  C_{def}(EMPLOYEE) = (**AND** EMPLOYEE (**ALL** article PUBLICATION)
  (**ALL** employer **Deptypes** ∪ {restypes}))

  Deptypes is a type defined in the database. The symbols restypes, employer and article are taken from the ontology. The definition context (defined later in this section) expresses an association between EMPLOYEE and PUBLICATION which may not be captured in the database.
- V_i can be a variable associated with a context.
  - This can be used to express constraints which the result of a query should obey. This is called the constraint context and is defined later in this section.

– The constraints would apply to the set, type or object the variable X would unify with.

Example:
Suppose we want all the articles whose titles contain the substring "abortion" in them. This can be expressed in the following query context:
C_q = <(article, X∘ <(title, {y|substring(y) = "abortion"})>)> = <(article, X∘Cntxt)>
where ∘ denotes association of a context with a variable and
Cntxt = <(title, {y|substring(y) = "abortion"})>
Association of a variable and a context ensures that the answer satisfies the constraints expressed in the context.
The same thing can be expressed in a DL as follows:
C_q = (**AND** ANSWER (**FILLS** article (**ALL** title {y|substring(y) = "abortion"})))

- V_i can be a set, type or an object associated with a context.
  - This is called the association context and is defined later in this section.
  - This may be used to express semantic dependencies between objects which may not be modeled in the database.

Example:
Suppose we want to represent information relating publications to employees in a database. Let PUBLICATION and EMPLOYEE be objects in a database. The definition context of HAS-PUBLICATION can be defined as:
C_{def}(HAS-PUBLICATION) = <(article, PUBLICATION)
(author, EMPLOYEE∘ <(affiliation, {research})>)>
C_{def}(HAS-PUBLICATION) = <(article, PUBLICATION) (author, EMPLOYEE∘Cntxt)>
where ∘ denotes association of a context with an object and Cntxt = <(affiliation, {research})>
Association of a context with an object is similar to defining a view on the object extensions such that only those instances satisfying the constraints defined in the context are exported to the federation. The same thing can be expressed in a DL as follows:
C_{def}(HAS-PUBLICATION) = (**AND** HAS-PUBLICATION
(**ALL** article PUBLICATION)
(**ALL** author (**AND** EMPLOYEE
(**ALL** affiliation (**ONE-OF** {research})))))

Note that the relationships between EMPLOYEE, PUBLICATION and HAS-PUBLICATION is information represented in the context not modeled in the database.

3.2.1 Definition Context of an Object

Given an object O in a database and a collection of contextual coordinates C_is from the ontology, the definition context is denoted as C_{def}(O) and can be used in the following ways:

- To specify the assumptions used in the design of the object O.
- To share only a pre-determined extension of the object O with the federation of databases. This exported object is denoted as O_F.

The associations between the objects stored in the database and the objects exported to the federation are expressed using the concepts of **semantic proximity** and **schema correspondences** (defined in Section 6.1).

3.2.2 Association Context of Objects

Given objects O and O_1 in a database the dependence of the definition context of O on the context of association between O and O_1, $\mathbf{C_{ass}(O_1, O)}$ can be represented as:

$C_{def}(O) = <(C_1, O_1 \circ C_{ass}(O_1, O)) \ldots (C_k, V_k) >$

The association context can be used in the following ways:

- To represent relationships between two objects with reference to an aspect of an application domain. This is done by associating it with the appropriate contextual coordinate.
- Different relationships between two objects may hold with reference to different aspects of the subject domain. This can be modeled by different association contexts between the two objects associated with different contextual coordinates.
- To model the relationships between the object O and different (more than one) objects as a part of the definition context of the same object. Thus, the context of an object would consist of its relationships with other objects.

3.2.3 Query Context

Whenever a query Q is posed to a federation of databases, we associate with it a query context $\mathbf{C_q}$ which makes explicit the partial semantics of the query Q.

- The user can consult ontologies to construct the query context in a semi-automatic manner. Issues of combining and displaying ontologies to enable a user to do this easily are discussed in [KS94a, KS95a].
- Objects and types defined in databases are also available to the user by relating them to some concept in an ontology.
- The query is expressed as a set of constraints which an answer object must satisfy. The constraints expressed in the query context can express incomplete information.

3.2.4 Constraint Context

The constraint context, C_{constr}(X,ANSWER) is typically a part of the query context and is used to pose constraints on the answer returned for the query.

$C_q = <(C_1, X \circ C_{constr}(X, ANSWER)) \ldots (C_k, V_k)>$

- It is associated with a variable which may be a place-holder for the answer or a part of the answer. The variable may be instantiated to an object or type definition.
- The context may represent constraints on the object and its attributes or the contextual coordinates associated with an object.
- The constraints which we currently limit to are cardinality constraints on sets and those that may be defined as a predicate on the elements of a set.

3.3 Reasoning about and manipulation of contexts

We have proposed a partial representation of context in the previous section. To use this representation meaningfully to focus on relevant information and to correlate information the following needs to be precisely defined:

- The most common relationship between contexts is the "specificity" relationship. Given two contexts C_1 and C_2, $C_1 \leq C_2$ iff C_1 is at least as specific as C_2. This is useful when objects defined in a particular context have to transcend [McC93] to a more specific or general context. This is discussed in detail with examples in [KS95b].
- It is also the case that two contexts may not be comparable to each other, i.e. it may not be possible to decide whether one is more general than the other or not. Thus, the specificity relationship gives us a partial order.
- For every two contexts we define the greatest lower bound of two contexts as the most specific context which is more general than each of the two contexts. The set of contexts thus forms a meet semi-lattice.

3.3.1 The specificity relationship

The specificity relationship between two contexts determines which context is more general than the other. We have defined this relationship with the help of specificity rules governing the contextual coordinates and their values.

Let $\text{Cntxt}_1 = <(C_1, V_1)\ (C_2, V_2)\ ...\ (C_k, V_k)>$
$\text{Cntxt}_2 = <(C'_1, V'_1)\ (C'_2, V'_2)\ ...\ (C'_m, V'_m)>$

$\text{Cntxt}_1 \leq \text{Cntxt}_2$ iff Cntxt_1 is **at least as specific as** Cntxt_2

In the following exposition, C, C_1, C_2, C'_1, C'_2, ... denote the contextual coordinates of the contexts under consideration. V, V_1, V_2, V'_1, V'_2, ... denote the values of the contextual coordinates. A, A_1, A_2, ..., S, S_1, S_2, ... stand for sets. X, Y, Z, ... stand for variables.

The specificity rules for the values of the contextual coordinates (V_is) are as follows:

Variable Specificity: $V_1 \leq X$, anything is more specific than a variable

Set Specificity: $S_1 \leq S_2$ iff $S_1 \subseteq S_2$

Association Context Specificity: These are rules concerning specificity of contextual coordinates when an association context is involved.

- $A_1 \circ \text{Cntxt}_i \leq A_2$ iff $A_1 \leq A_2$
- $A_i \circ \text{Cntxt}_i \leq A_j \circ \text{Cntxt}_j$ iff $A_i \leq A_j \wedge \text{Cntxt}_i \leq \text{Cntxt}_j$

$\text{Cntxt}_1 \leq \text{Cntxt}_2$ if the following conditions hold:

- $m \leq k$
- $\forall$i, $1 \leq i \leq m$, $\exists$j $C_j \leq C'_i$[2] $\wedge$ $V_j \leq V'_i$

3.3.2 Operations on the Context Lattice

As observed earlier, the specificity relationship between the contexts induces a partial order among the contexts. Thus the context can be organized as a meet semi-lattice where every pair of contexts has the greatest lower bound. In this subsection we define the *glb* operation and other operations which we will use later in the paper.

overlap(Cntxt_1, Cntxt_2) $= \{\ C_i |\ C_i \in \text{Cntxt}_1 \wedge C_i \in \text{Cntxt}_2\ \}$

coherent(Cntxt_1, Cntxt_2) This operator determines whether the constraints determined by the values of the contextual coordinates are consistent.

Example:
Let $\text{Cntxt}_1 = <(\text{salary}, \{x |\ x \leq 10000\})>$
$\text{Cntxt}_2 = <(\text{salary}, \{x |\ x > 10000\})>$
Thus, coherent(Cntxt_1, Cntxt_2) = FALSE

The glb of two Contexts

We now define the greatest lower bound of two contexts with the help of the rules that determine the greatest lower bounds of the contextual coordinates and their values. The rules determining **glb(V_i, V'_j)** are:

Variable: $\text{glb}(V_i, X) = V_i$

Sets: $\text{glb}(S_1, S_2) = S_1 \cap S_2$

Association Contexts: These are rules concerning the glb of the values of the contextual coordinates when an association context is involved.

[2] This specificity relationship between contextual coordinates is determined from the ontology and is beyond the scope of this paper. In defining the various operations on the context lattice we shall use the equality comparison instead.

- glb($A_1 \circ Cntxt_i$, A_2) = glb(A_1, A_2)$\circ Cntxt_i$
- glb($A_i \circ Cntxt_i$, $A_j \circ Cntxt_j$) = glb(A_i, A_j)$\circ$glb($Cntxt_i$, $Cntxt_j$)

The greatest lower bound of the contexts **glb($\mathbf{Cntxt_1}$, $\mathbf{Cntxt_2}$)** can now be defined as:

- glb($Cntxt_1$, $Cntxt_2$) = $Cntxt_1$, if $Cntxt_2$ = <> *[Empty Context]*
- (C_i, V_i) $\in$ glb($Cntxt_1$, $Cntxt_2$), if $C_i \notin$ overlap($Cntxt_1$, $Cntxt_2$)
- (C'_i, V'_i) $\in$ glb($Cntxt_1$, $Cntxt_2$), if $C'_i \notin$ overlap($Cntxt_1$, $Cntxt_2$)
- (C_k, glb(V_k, V'_j)) $\in$ glb($Cntxt_1$, $Cntxt_2$), if $C_k = C'_j \in$ overlap($Cntxt_1$, $Cntxt_2$)

An alternative equivalent representation of a context (expressed using the glb operation) is very useful when there is a need to carry out inferences on the context and information associated with it.

$Cntxt = <(C_1, V_1)(C_2, V_2) \ldots (C_k, V_k)>$
$= glb(<(C_1, V_1)>, glb(<(C_2, V_2)>, \ldots, glb(<(C_k, V_k)>, <>) \ldots))$

Example:

Consider the following two contexts:

$Cntxt_1$ = <(author, EMPLOYEE$\circ$ <(affiliation, {research})>) (article, PUBLICATION)>

$Cntxt_2$ = <(article, X$\circ$ <(title,{x| substring(x)="abortion"})>)>

It should be noted that:

- article $\in$ overlap($Cntxt_1$,$Cntxt_2$)
  $\Rightarrow$ (article, glb(PUBLICATION, X$\circ$ <(title, {x|substring(x) = "abortion"})>))
  $\in$ glb($Cntxt_1$, $Cntxt_2$)
- author $\notin$ overlap($Cntxt_1$,$Cntxt_2$) $\Rightarrow$
  (author, EMPLOYEE$\circ$ <(affiliation, {research})>) $\in$ glb($Cntxt_1$, $Cntxt_2$)
- glb(PUBLICATION, X$\circ$ <(title, {x|substring(x) = "abortion"})>)
  = glb(PUBLICATION,X)$\circ$ <(title,{x|substring(x) = "abortion"})>
  [Association Contexts]
  = PUBLICATION$\circ$ <(title,{x|substring(x) = "abortion"})>
  [glb of a variable and an object]

glb($Cntxt_1$,$Cntxt_2$) = <(author, EMPLOYEE$\circ$ <(affiliation, {research})>)
(article, PUBLICATION$\circ$ <(title,{x| substring(x)="abortion"})>)>

3.4 Issues of language and ontology in context representation

In this section we discuss the issues of a language in which the explicit representation discussed above can be best expressed. We also discuss issues of ontology, i.e. the vocabulary used by the language to represent the contexts.

3.4.1 Language for context representation

In Section 3.2 we have proposed a context representation as a collection of contextual coordinates and their values. The values themselves may have contexts associated with them. In this section, we enumerate the properties desired of a language to express the context representation.

- The language should be declarative in nature as the context shall typically be used to express constraints on objects in an intensional manner. Besides, the declarative nature of the language will make it easier to perform inferences on the context.
- The language should be able to express the context as a collection of contextual coordinates, each describing a specific aspect of information present in the database or requested by a query.

- The language should have primitives (for determining the subtype of two types, pattern matching, etc.) in the model world, which might be useful in comparing and manipulating context representations.
- The language should have primitives to perform navigation in the ontology to identify the abstractions related to the ontological objects in the query context or the definition contexts of objects in the databases.

3.4.2 The Ontology Problem

In constructing the contexts as illustrated in Section 3.2 the choice of the contextual coordinates (C_is) and the values assigned to them (V_is) is very important. There should be *ontological commitments*, i.e. agreements about the ontological objects used between the users and the information system designers. In our case this corresponds to an agreement on the terms used for the contextual coordinates and their values by a user in formulating the query context and a database administrator for formulating the definition and association contexts. In an example in Section 3.2, we have defined C_{def}(EMPLOYEE) by making use of symbols like *employer, affiliation* and *reimbursement* from the ontology for contextual coordinates and *research, teaching* etc. for the values of the contextual coordinates.

We assume that each database has available to it an ontology corresponding to a specific domain. The definition and association contexts of the objects take their terms and values from this ontology. However in designing the definition contexts and the query context, the issues of combining the various ontologies arise.

We now enumerate various approaches one might take in building ontologies for a federation of information sources. Other than the ontological commitment, a critical issue in designing ontologies is the **scalability** of the ontology as more information sources enter the federation.

- **The Common Ontology approach:**
  - One approach has been to build an extensive global ontology. A notable example of global ontology is Cyc [LG90] consisting of around 30,000 objects. In Cyc, the mapping between each individual information resource and global ontology is accomplished by a set of *articulation axioms* which are used to map the entities of an information resource to the concepts (such as frames and slots) in Cyc's existing ontology [CHS91].
  - Another approach has been to exploit the semantics of a single problem domain (e.g., transportation planning) [ACHK93]. The domain model is a declarative description of the objects and activities possible in the application domain as viewed by a typical user. The user formulates queries using terms from the application domain.
- **Re-use of Existing Ontologies:** Given our assumption that there will be numerous information systems participating in the federation, it is unrealistic to expect any one existing ontology or classification to suffice. We propose a re-use of various existing classifications such as ISBN classification for publications, botanical classification for plants etc. An example of such a classification is illustrated in Figure 3. These ontologies can then be combined in different ways and made available to the federation.
  - A critical issue in combining the various ontologies is determining the overlap between them. One possibility [Wie94] is two define the "intersection" and "mutual exclusion" points between the various ontologies.
  - Another approach has been adopted in [MS95]. The types determined to be similar by a sharing advisor are classified into a collection called *concept*. A *concept hierarchy* is thus generated modeling superconcept-subconcept relationship. These types may be from different databases and their similarity or dissimilarity is based on heuristics with user input as required.

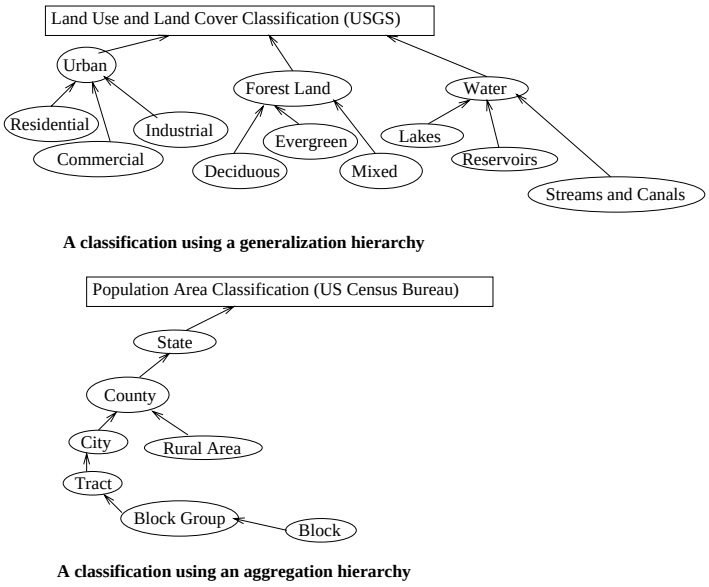

Figure 3: Examples of Generalization and Aggregation hierarchies for Ontology construction

4 A Semantic Taxonomy

Our emphasis is on identifying semantic similarity even when the objects have significant representational differences [She91]. Semantic proximity is an attempt to characterize the degree of semantic similarity between two objects using the RWS. It provides a qualitative measure to distinguish between the terms introduced in [She91], such as *semantic equivalence*, *semantic relationship*, *semantic relevance* and *semantic resemblance*. Two objects can be semantically related in one of the above four ways. Semantic equivalence is *semantically closer* than semantic relationship, and so on.

In this section we use the concept of semantic proximity defined in Section 2 and the context representation discussed above to define a semantic taxonomy consisting of the various types of semantic similarities between object. The taxonomy thus designed, is illustrated in Figure 5.

4.1 The role of context in semantic classification

The context, which is the pivot on which the semantic proximity depends, plays a key role in this taxonomy. Here we enumerate the possible values for context.

- ALL, i.e., the semPro between the objects is being defined *wrt* all known and *coherent* comparison contexts. There should be coherence between the definition contexts of the objects being compared and between the definition contexts and the context of comparison.
- SOME, i.e., the semPro between the objects is being defined *wrt* some context. This context may be constructed in the following ways.
  - GLB, i.e. the greatest lower bound of the contexts of the two objects Typically we are interested in the *glb* of the context of comparison and the definition context of the object.
  - LUB, i.e. the least upper bound[3] of the contexts of the two objects is taken. Typically, we are interested in the *lub* of the definition contexts of the two objects when there does not exist an abstraction/mapping between their domains in the context of comparison.

[3] We have not defined it for the general case. Here, we are only interested in the special case: $(C_k, V_k \cup V'_j) \in \text{lub}(\text{Cntxt}_1, \text{Cntxt}_2)$ where $C_k = C'_j \in \text{overlap}(\text{Cntxt}_1, \text{Cntxt}_2)$

- SUB-CONTEXTS, we might be interested in the **semPro** between two objects in contexts which are more specific or more general *wrt* the context of comparison.
- NONE, i.e. there doesn't exist a context in which a meaningful abstraction or mapping between the domains of the objects may be defined. This is the case when the definition contexts of the objects being compared are *not coherent* with each other.

4.2 Semantic Equivalence

This is the strongest measure of semantic proximity two objects can have. Two objects are defined to be *semantically equivalent* when they represent the same real world entity or concept. Expressed in our model, it means that given two objects O_1 and O_2, it should be possible to define a total 1-1 value mapping between the domains of these two objects in any known and coherent context. Thus we can write it as:

semPro(O_1, O_2) = <ALL, total 1-1 value mapping, (D_1, D_2), _>[4]

The notion of equivalence described above depends on the definition of the domains of the objects and can be more specifically called *domain semantic equivalence*. We can also define a stronger notion of semantic equivalence between two objects which incorporates the state of the databases to which the two objects belong. This equivalence is called *state semantic equivalence* and is defined as:

semPro(O_1, O_2) = <ALL, M, (D_1, D_2), (S_1, S_2) >
where M is a total 1-1 value mapping between (D_1, S_1) and (D_2, S_2).

For the purposes of this paper we shall use semantic equivalence to mean domain semantic equivalence.

4.3 Semantic Relationship

This type of semantic similarity is weaker than semantic equivalence. Two objects are said to be *semantically related* when there exists a partial many-one value mapping, or a generalization, or aggregation abstraction between the domains of the two objects. Here we relax the requirement of a 1-1 mapping in a way that given an instance O_1 we can identify an instance of O_2 but not vice versa. The requirement that the mapping be definable in all the known and coherent contexts is not relaxed. Thus we define the *semantic relationship* as:

semPro(O_1, O_2) = <ALL, M, (D_1, D_2), _>
where M may be a partial many-one value mapping, generalization, or aggregation

4.4 Semantic Relevance

We consider two objects to be *semantically relevant* if they can be related to each other using some *abstraction* in *some context*. Thus the notion of semantic relevance between two objects is context dependent, i.e., two objects may be semantically relevant in one context, but not so in another. Objects can be related to each other using any abstraction.

semPro(O_1, O_2) = <SOME, ANY, (D_1, D_2), _>

4.5 Semantic Resemblance

This is the weakest measure of semantic proximity, which might be useful in certain cases. Here, we consider the case where the domains of two objects cannot be related to each other by any abstraction in any context. Hence, the exact nature of semantic proximity between two objects is very difficult to specify. In this case, the user may be presented with extensions of both the objects. In order to express this type of semantic similarity, we introduce an aspect of context, which we call **role**, by extending the concept of role defined in [EN89]. Semantic resemblance is defined in detail in the next section.

[4] We use the "_" sign to denote don't care.

4.5.1 Role played by an Object in a Context

This refers to the relationship between an object and the semantic context to which it belongs. We characterize this relationship as a binary function, which has the object and its context as the arguments and the name of the role as the value.

$$\text{role-of} : \text{object} \times \text{context} \rightarrow \text{rolename}$$

The mapping defined above may be multi-valued, as it is possible for an object to have multiple roles in the same context.

Based on the representation of a context proposed in Section 3.2 we can express this by constructing the least upper bound of the contexts. Consider the type **Number** and the type **Name** defined in the databases.

C_{def}(Database1) = <(Class, {Employee, ... }) (Identifier, {Name, ...})>
C_{def}(Database2) = <(Class, {Employee, ... }) (Identifier, {Number, ...})>
lub(C_{def}(Database1), C_{def}(Database2))
= <(Class, {Employee_1, Employee_2, ...}) (Identifier, {Name, Number, ... })>

Thus role-of(Name, C_{def}(Database1)) = role-of(Number, C_{def}(Database2)) = Identifier
Since Name, Number $\in$ Identifier $\wedge$ Identifier $\in$ lub(C_{def}(Database1), C_{def}(Database2))
This is illustrated in Figure 4.

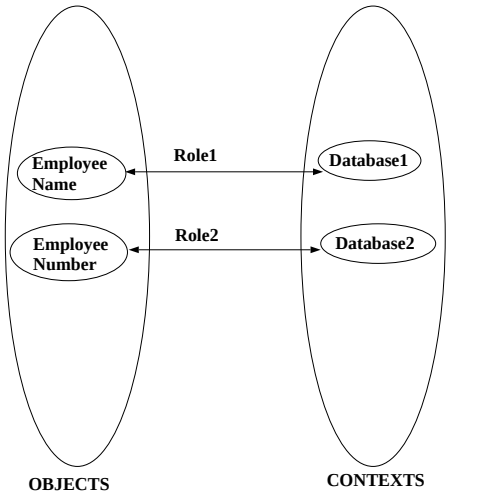

Figure 4: Roles played by objects in their contexts

4.5.2 Roles and Semantic Resemblance

Whenever two objects cannot be related to each other by any abstraction in any context, but they are associated with contexts in which they have the same role and their definition contexts are coherent wrt each other, they can be said to *semantically resemble* each other. This is a generalization of DOMAIN-DISJOINT-ROLE-EQUAL concept in [LNE89].

semPro(O_1, O_2) = <SOME(LUB), NONE, (D_1, D_2), _>
where coherent(C_{def}(O_1),C_{def}(O_2)) and $\exists \text{Cntxt}_1$,Cntxt_2 exported by DB_1,DB_2 respectively
and SOME(LUB) denotes a context defined as follows:

context = lub($Cntxt_1$, $Cntxt_2$) and $D_1 \neq D_2$
and role-of(O_1, context) = role-of(O_2, context)

4.6 Semantic Incompatibility

While all the qualitative proximity measures defined above describe semantic similarity, semantic incompatibility asserts semantic dissimilarity. Lack of any semantic similarity does not automatically imply that the objects are semantically incompatible. Establishing semantic incompatibility requires asserting that the definition contexts of the two objects are *incoherent wrt* each other and there do not exist contexts associated with these objects such that they have the same role.

semPro(O_1, O_2) = <NONE, NONE, (D_1, D_2), _>
where $C_{def}(O_1)$ and $C_{def}(O_2)$ are incoherent with each other
and D_1 may or may not be equal to D_2
and $\nexists$ context such that role-of(O_1, context) = role-of(O_2, context)

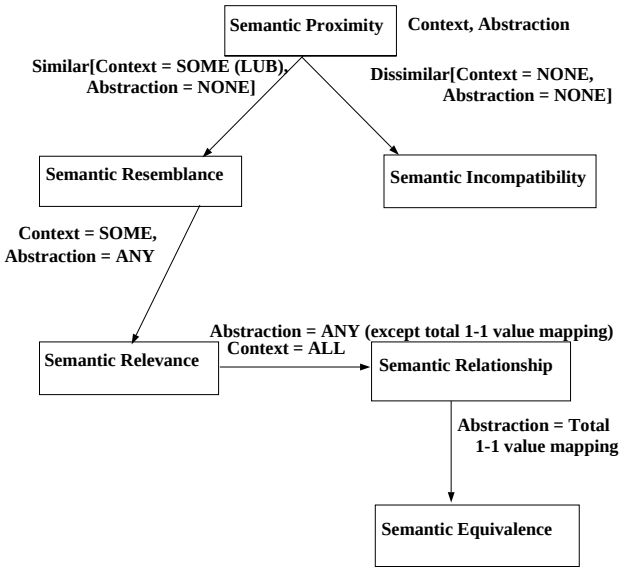

Figure 5: Semantic Classification of Object Similarities

5 Schematic Heterogeneities in Multidatabases

In this section we deal with a broad class of schematic differences and the possible semantic similarities between objects having schematic differences [SK92]. With each type of schematic difference, we enumerate the possible semantic proximity descriptors. The broad classes of schematic heterogeneities we are dealing with are: *domain incompatibility*, *entity definition incompatibility*, *data value incompatibility*, *abstraction level incompatibility* and *schematic discrepancies* (Figure 6). While the issue of schematic/representational/structural heterogeneity have been dealt with by a number of researchers [DH84, BOT86, CRE87, KLK91, KS91], the unique feature of our work is the strong correlation between the semantic aspects defined above and the structural aspects.

5.1 Domain Incompatibility

In this section we discuss the incompatibilities that arise when two different domain types are used as different definitions of semantically similar attribute domains. We refine the broad definition of this incompatibility given in [CRE87]. We also discuss the possible semantic similarities with each case (Figure 7).

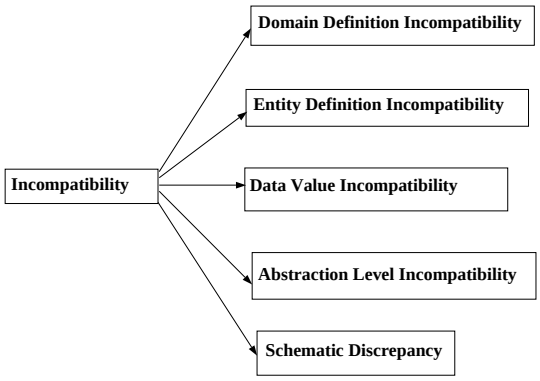

Figure 6: Schematic Heterogeneities

5.1.1 Naming Conflicts

Two attributes that are semantically alike might have different names. They are known as *synonyms*.

Example:

Consider two databases having the relations:

```
STUDENT(Id#, Name, Address)
TEACHER(SS#, Name, Address)
```

`Id# of STUDENT and SS# of TEACHER are synonyms.`

Mappings between synonyms can often be established *wrt* all known and coherent contexts. In such cases, the two domain types may be considered *semantically equivalent*.

Two attributes that are semantically unrelated might have the same names. They are known as *homonyms*.

Example:

Consider two databases having the relations:

```
STUDENT(Id#, Name, Address)
BOOK(Id#, Name, Author)
```

`Id# of STUDENT and BOOK are homonyms.`

The definition contexts of the two domain types (which are defined in two different databases) may be modeled as follows:

C_{def}(STUDENT.Id#) = <(identifies, AnimateObject)>
C_{def}(BOOK.Id#) = <(identifies, InAnimateObject)>
The concepts AnimateObject and InAnimateObject are obtained from an ontology for the domain.

Since homonyms are semantically unrelated, their definition contexts are modeled in a way that they are *incoherent wrt* each other. Thus these two domain types may be considered *semantically incompatible*.

5.1.2 Data Representation Conflicts

Two attributes that are semantically similar might have different data types or representations.

Example:

`STUDENT.Id# is defined as a 9 digit integer.`

`TEACHER.SS# is defined as an 11 character string.`

Conversion mappings or routines between different data representations can often be established *wrt* all known and coherent contexts. In such cases, these domain types may be considered *semantically equivalent*.

5.1.3 Data Scaling Conflicts

Two attributes that are semantically similar might be represented using different units and measures. There is a one-one mapping between the values of the domains of the two attributes. For instance, the salary attribute might have values in $ and £.

Typically mappings between data represented in different scales can be easily expressed in terms of a function or a lookup table, or by using dynamic attributes as in [LA86] and *wrt* all known and coherent contexts. In such cases, the domain types may be considered *semantically equivalent*.

5.1.4 Data Precision Conflicts

Two attributes that are semantically similar might be represented using different precisions. This case is different from the previous case because there may not be one-one mapping between the values of the domains. There may be a many-one mapping from the domain of the precise attribute to the domain of the coarser attribute.

Example:

Let the attribute Marks have an integer value from 1 to 100.
Let the attribute Grades have the values {A, B, C, D, F}.

Marks	Grades
81-100	A
61-80	B
41-60	C
21-40	D
1-20	F

Table 2: Mapping between Marks and Grades

There may be a many-one mapping from Marks to Grades (Table 2). Grades is the coarser attribute. Typically, mappings can be specified from the precise data scale to the coarse data scale *wrt* all known and coherent contexts. Given a letter grade, determining the precise numerical score is typically not possible. In such cases, the domain types may be considered *semantically related*.

5.1.5 Default Value Conflicts

This type of conflict depends on the definition of the domain of the concerned attributes. The *default value* of an attribute is that value which it is defined to have in the absence of more information about the real world. For instance, the default value for Age of an adult might be defined as 18 years in one database and as 21 years in another.

It may not be possible to specify mappings between a default value of one attribute to the default value of another in all known and coherent contexts. However, it is often possible to do so *wrt* some context. In such cases, the domain types can be considered to be *semantically relevant*, i.e., their *semantic proximity* can be defined as follows:

semPro(Age_1, Age_2) = <SOME, Abstraction, (D_1, D_2), _>
Context = <(default, DefaultAge)>
where the concept DefaultAge is obtained from an ontology for the domain. When the semPro is *evaluated* wrt the Context, it maps to different ages in the different databases.

5.1.6 Attribute Integrity Constraint Conflicts

Two semantically similar attributes might be restricted by constraints which might not be consistent with each other. For instance, in different databases, the attribute Age might follow these constraints:

Example:
C1 : $Age_1 \leq 18$
C2 : $Age_2 > 21$
C1 and C2 are inconsistent and hence the integrity constraints on the attribute Salary are said to conflict.

If the constraints are captured in the definition contexts of the domain types of Age_1 and Age_2, then they would be incoherent and can be considered *semantically incompatible*. However, in the case these types are play the same role in the definition contexts of their respective databases in which they exist, they may be considered to have a *semantic resemblance* to each other.

$C_{def}(Database_1)$ = <(timePeriod, {Age, Duration, ...})>
$C_{def}(Database_2)$ = <(timePeriod, {Age, RacePerformance, ...})>
where Age_1, Age_2 denote the attribute Age in $Database_1$, $Database_2$ respectively

semPro(Age_1, Age_2) = <SOME(LUB), NONE, (D_1, D_2), _>
where SOME(LUB) denotes a context defined as follows:
where context = lub($C_{def}(Database_1)$, $C_{def}(Database_2)$) and $D_1 \neq D_2$
and role-of(Age_1, context) = role-of(Age_2, context) = timePeriod.

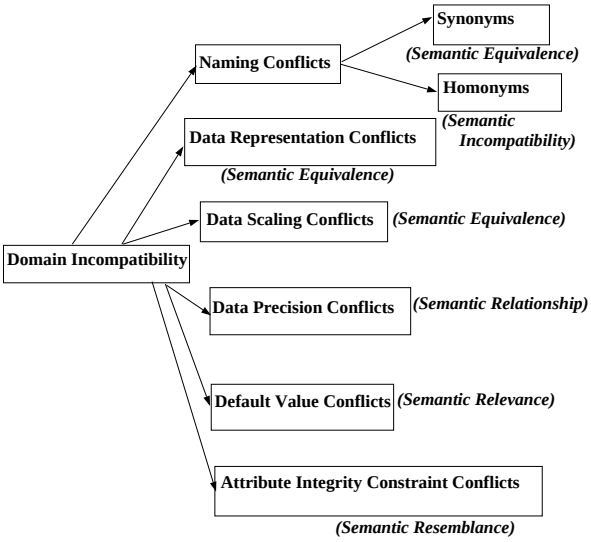

Figure 7: Domain Incompatibility and the likely types of semantic proximi ties

5.2 Entity Definition Incompatibility

In this section we discuss the incompatibilities that arise between two objects when the entity descriptors used by the objects are only partially compatible, even when the same type of entity is being modeled. We refine the broad definition of this class of conflicts given in [CRE87]. We also discuss the possible semantic similarities with each case (Figure 8).

5.2.1 Database Identifier Conflicts

In this case, the entity descriptions in two databases are incompatible because they use identifier records that are semantically different.

Example:

```
                STUDENT1(SS#, Course, Grades)
                STUDENT2(Name, Course, Grades)
STUDENT1.SS# and STUDENT2.Name are semantically different keys.
```

The semantic proximity of objects having this kind of conflict depends on whether it is possible to define an abstraction to map the keys in one database to another. However, if we assume that the context(s) of the identifiers are defined in the local schemas, we know that they play the *role* of *identification* in their respective contexts. Hence, the weakest possible measure of *semantic resemblance* applies, though stronger measures might apply too.

semPro(SS#, Name) = <SOME(LUB), _ , (D_1, D_2), _>
where D_1 = Domain(SS#) and D_2 = Domain(Name)
and where SS# and Name exist in $Database_1$ and $Database_2$ respectively
$C_{def}(Database_1)$ = <(Class, {STUDENT1, ... }) (Identifier, {SS#, ...})>
$C_{def}(Database_2)$ = <(Class, {STUDENT2, ... }) (Identifier, {Name, ...})>
and SOME(LUB) denotes a context defined as follows:
and context = lub($C_{def}(Database_1)$, $C_{def}(Database_2)$)
and role-of(SS#, context) = role-of(Name, context) = Identifiers

5.2.2 Naming Conflicts

Semantically alike entities might be named differently in different databases. For instance, EMPLOYEE and WORKERS might be two objects describing the same set of entities. They are known as *synonyms*. Typically, mappings between synonyms can often be established *wrt* all known and coherent contexts. In such cases the objects may be considered *semantically equivalent*.

On the other hand, semantically unrelated entities might have the same name in different databases. For instance, TICKETS might be the name of a relation which models movie tickets in one database, whereas it might model traffic violation tickets in another database. They are known as *homonyms* of each other. In a manner similar to that demonstrated in Section 5.1.1, their definition contexts can be modeled in a way that they are incoherent *wrt* each other. Thus these objects may be considered *semantically incompatible*.

5.2.3 Schema Isomorphism Conflicts

Semantically similar entities may have different number of attributes, giving rise to schema isomorphism conflicts.

Example:

```
INSTRUCTOR1(SS#, HomePhone, OffPhone)
INSTRUCTOR2(SS#, Phone)
```

`is an example of schema non-isomorphism.`

It should be noted that this can be considered an artifact of the *Data Precision Conflicts* identified in Section 5.1.4 of this paper, as the Phone number of INSTRUCTOR1 can be considered to be represented in a more precise manner than the Phone number of INSTRUCTOR2. However, the conflicts discussed in Section 5.1.4 are due to the differences in the domains of the attributes representing the same information and hence are *attribute level conflicts*. Whereas, conflicts in this sections arise due to differences in the way the entities INSTRUCTOR1 and INSTRUCTOR2 are defined in the two databases and hence are *entity level conflicts*.

Since mappings can be established between the objects on the basis of the common and identifying attributes, the two objects may be considered *semantically related*.

semPro($Instructor_1$, $Instructor_2$)
= <ALL, {M_{ID}, M_1}, ({$D_{1,SS\#}$, $D_{HomePhone}$, $D_{OffPhone}$}, {$D_{2,SS\#}$, D_{Phone}}), _>
where M_{ID} is a total 1-1 value mapping between $D_{1,SS\#}$ and $D_{2,SS\#}$ and represents the mapping between the identifiers of the two objects.
M_1 may be a total/partial 1-1/many-one value mapping between $D_{HomePhone} \cup D_{OffPhone}$ and D_{Phone}.

5.2.4 Missing Data Item Conflicts

This conflict arises when of the entity descriptors modeling semantically similar entities, one has a missing attribute. This type of conflict is subsumed by the conflict discussed in the previous section. A special case of the above conflict which satisfies the following conditions:

- The missing attribute is compatible with the entity, and
- There exists an inference mechanism to deduce the value of the attribute.

Example:

```
                STUDENT(SS#, Name, Type)
                GRAD-STUDENT(SS#, Name)
STUDENT.Type can have values "UG" or "Grad"
GRAD-STUDENT does not have a Type attribute, but that can be implicitly
deduced to be "Grad".
```

In the above example, GRAD-STUDENT can be thought to have a Type attribute whose default value is "Grad". The conflict discussed in this section is different from the *default value* conflict in Section 5.1.5 which is an *attribute level conflict* whereas the conflict discussed here is an *entity level conflict*. The objects may be considered *semantically relevant* as proposed below.

The definition contexts of the two objects can be defined as:
C_{def}(STUDENT) = <(type, {graduate, undergraduate})>
C_{def}(GRAD-STUDENT) = <(type, {graduate})>

The context in which semPro(STUDENT, GRAD-STUDENT) will be defined as:

glb(C_{def}(STUDENT), C_{def}(GRAD-STUDENT)) = <(type, {graduate})>
The abstraction is then computed by "conditioning" the original student abstraction *wrt* this new context. Since every abstraction/mapping is associated with a context, the change in the abstraction as a result of the change in the associated context is called conditioning and is discussed in detail in [KS95b].

semPro(STUDENT, GRAD-STUDENT) = <SOME, M, (D_1, D_2), _>
where M: STUDENT → GRAD-STUDENT is a partial 1-1 value mapping
and Context = SOME = <(type, {graduate})>

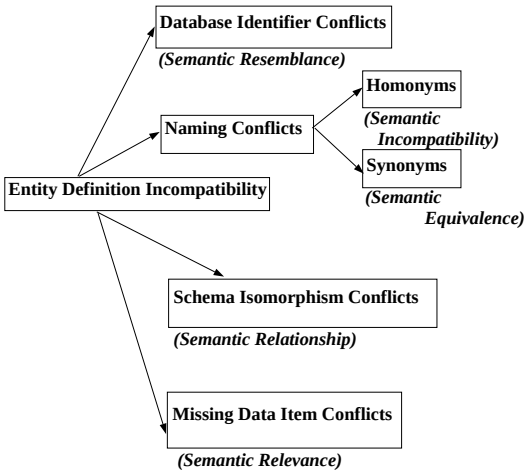

Figure 8: Entity Definition Incompatibilities and the likely types of semantic proximities

5.3 Data Value Incompatibility

This class of conflicts covers those incompatibilities that arise due to the values of the data present in different databases [BOT86]. These conflicts are different from default value conflicts (Section 5.1.5) and attribute integrity constraint conflicts (Section 5.1.6) in that the latter are due to the differences in the definitions of the domain types of the attributes. Here we refer to the data values already existing in the database. Thus, the conflicts here depend on the database state. Since we are dealing with independent databases, it is not necessary that the data values for the same entities in two different databases be consistent with each other. We also discuss the possible semantic similarities with each case (Figure 9).

Example:

```
Consider two databases modeling the entity Ship
                    SHIP1(Id#, Name, Weight)
                    SHIP2(Id#, Name, Weight)
Consider a entity represented in both databases as follows :
                    SHIP1(123, USSEnterprise, 100)
                    SHIP2(123, USSEnterprise, 200)
Thus, we have the same entity for which SHIP1.Weight is not the same as
SHIP2.Weight, i.e., it has inconsistent values in the database.
```

5.3.1 Known Inconsistency

In this type of conflict, the cause of inconsistency is known ahead of time and hence measures can be initiated to resolve the inconsistency in the data values. For instance, it might be known ahead of time that one database is more reliable than the other. This information can typically be represented in the query context C_q. Here the similarity of objects depends on the state component of semPro and are hence considered *state semantically relevant.*

C_q = <(class, SHIP) (dataItem, {Id#}) (choose-from, {DB1})>

semPro(O_1, O_2) = <C_q, M, (D_1, D_2), (S_1, S_2)>
where M is a total 1-1 value mapping between (D_1, S_1) and (D_2, S_2)
(In this case the default is (D_1, S_1)).

5.3.2 Temporary Inconsistency

In this type of conflict, the inconsistency is of a temporary nature. This type of conflict has been identified in [RSK91] and has been expressed as a *temporal consistency predicate*[5]. One of the databases which has conflicting values, might have obsolete information. This means that the information stored in the databases is time dependent. The time lag information (Δt) can be easily represented in the query context C_q and hence the objects may be considered *state semantically relevant.* The semPro when evaluated *wrt* context gives the mapping defined below.

C_q = <(class, SHIP) (dataItem, {Weight}) (timeLag, Δt)>
Here we model the state of an object as a function of time.
semPro(O_1, O_2)=<C_q, total 1-1 value mapping, (D_1, D_2), (S_1, S_2)>
where $S_2(t + \Delta t) = S_1(t)$.

5.3.3 Acceptable Inconsistency

In this type of conflict, the inconsistencies between values from different databases might be within an acceptable range. Thus, depending on the type of query being answered, the error in the values of two inconsistent databases might be considered tolerable. The *tolerance* of the inconsistency can be of a numerical or non numerical nature and can be easily represented in

[5] Additional information on weaker criteria for consistency can be found in the literature on transaction models (e.g., see [SRK92]).

the query context C_q and hence the objects may be considered *state semantically relevant.*
Example: Numerical Inconsistency
QUERY: Find the Tax Bracket of an Employee.
INCONSISTENCY: If the inconsistency in the value of an Employee Income is up to a fraction of a dollar it may be ignored.
C_q = <(class, EMPLOYEE) (dataItem, {Salary}) (epsValue, [0, 0.99])>
where epsValue is a contextual coordinate which models the level of inconsistency that can be tolerated for the query.
Example: Non numerical Inconsistency
QUERY: Find the State of Residence of an Employee.
INCONSISTENCY: If the Employee is recorded as staying in Edison and New Brunswick (both are in New Jersey), then again the inconsistency may be ignored.
C_q = <(class, EMPLOYEE) (dataItem, {Residence}) (epsValue, sameState)>

semPro(O_1, O_2)=<C_q, partial many-one value mapping, (D_1, D_2), (S_1, S_2)>
where perturb(S_1, ϵ) = S_2
and ϵ is the discrepancy in the state of the two objects.

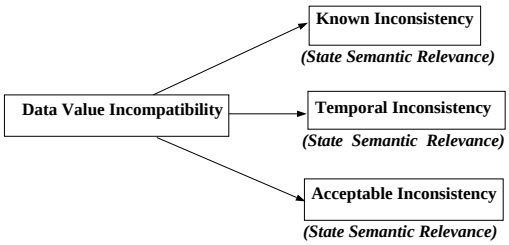

Figure 9: Data value incompatibilities and the likely types of semantic proximities

5.4 Abstraction Level Incompatibility

This class of conflicts was first discussed in [DH84] in the context of the functional data model. These incompatibilities arise when two semantically similar entities are represented at differing levels of abstraction. Differences in abstraction can arise due to the different levels of generality at which an entity is represented in the database. They can also arise due to aggregation used both at the entity as well as the attribute level. We also discuss the possible semantic similarities with each case (Figure 10).

5.4.1 Generalization Conflicts

These conflicts arise when two entities are represented at different levels of generalization in two different databases.
Example:

```
Consider the entity "Graduate Students" which may be
represented in two different databases as follows :
                STUDENT(Id#, Name, Major, Type)
                GRAD-STUDENT(Id#, Name, Major)
Thus we have the same entity set being defined at a more general
level in the first database.
```

The definition contexts of the two objects can be defined as:
C_{def}(STUDENT) = <(type, {graduate, undergraduate})>
C_{def}(GRAD-STUDENT) = <(type, {graduate})>

The context in which semPro(STUDENT, GRAD-STUDENT) is defined is given by:

glb(C_{def}(STUDENT), C_{def}(GRAD-STUDENT)) = <(type, {graduate})>
The abstraction is then computed by "conditioning" the original student abstraction *wrt* this new context. Thus, STUDENT and GRAD-STUDENT may be considered *semantically relevant.*

semPro(STUDENT, GRAD-STUDENT) = <SOME, M, (D_1, D_2), _>
where M: STUDENT → GRAD-STUDENT is a partial 1-1 value mapping
and Context = SOME = <(type, {graduate})>

5.4.2 Aggregation Conflicts

These conflicts arise when an aggregation is used in one database to identify a set of entities in another database. Also, the properties of the aggregate concept can be an aggregate of the corresponding property of the set of entities.

Example:

```
Consider the aggregation SET-OF which is used to define a concept in the
first database and the set of entities in another database as follows :
                    CONVOY(Id#, AvgWeight, Location)
                    SHIP(Id#, Weight, Location, Captain)
Thus, CONVOY in the first database is a SET-OF SHIPs in the second
database. Also, CONVOY.AvgWeight is the average(aggregate function)
of SHIP.Weight of ships that are members of the convoy.
```

In this case there is a mapping in only one direction, i.e., an element of a set is mapped to the set itself. In the other direction, the mapping is not precise. When the SHIP entity is known, one can identify the CONVOY entity it belongs to, but not vice versa. Also, the aggregation can be expressed in the definition context of CONVOY using the composition of contextual coordinates as follows:

C_{def}(CONVOY) = <(member, SHIP) (weight, ...) (location, ...)>
C_{def}(SHIP) = <(shipweight, ...) (shiplocation, ...)>
where weight = average(shipweight) and shiplocation = location are relationships between the various contextual coordinates obtained from the ontology of the domain.
context = glb(C_{def}(CONVOY), C_{def}(SHIP))

semPro(CONVOY, SHIP) = <context, Aggregation, (D_1, D_2), _>

Thus CONVOY and SHIP maybe considered *semantically relevant.*

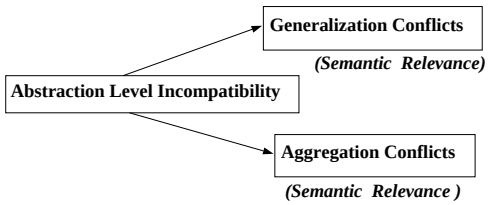

Figure 10: Abstraction level incompatibilities and the likely types of semantic proximities

5.5 Schematic Discrepancies

This class of conflicts was discussed in [DAODT85, KLK91]. It was noted that these conflicts can take place within the same data model and arise when data in one database correspond to metadata of another database. This class of conflicts is similar to that discussed in Section 5.3 when the conflicts depend on the database state. We now analyze the problem and identify

three aspects with help of an example given in [KLK91]. We also discuss the possible semantic similarities with each case (Figure 11).

Example:

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

- **Database DB1 :**
  relation r : {(date, stkCode, clsPrice) ... }
- **Database DB2 :**
  relation r : {(date, stk1, stk2, ...) ... }
- **Database DB3 :**
  relation stk1 : {(date, clsPrice) ... },
  relation stk2 : {(date, clsPrice) ... },
  ⋮

DB1 consists of a single relation that has a tuple per day per stock with its closing price. DB2 also has a single relation, but with one attribute per stock, and one tuple per day, where the value of the attribute is the closing price of the stock. DB3 has, in contrast, one relation per stock that has a tuple per day with its closing price. Let us consider that the stkCode values in DB1 are the names of the attributes, and in the other databases they are the names of relations (e.g., stk1, stk2).

5.5.1 Data Value Attribute Conflict

This conflict arises when the value of an attribute in one database corresponds to an attribute in another database. Thus, this kind of conflict depends on the *database state*. Referring to the above example, the values of the attribute *stkCode* in the database *DB1* correspond to the attributes *stk1, stk2*, ... in the database *DB2*.

The mappings here are established between sets of attributes ($\{O_i\}$) and values in the extension of the other attribute (O_2). This is possible, however only *wrt* the contexts of the databases they are in. The two objects model data at different levels and hence may be considered to be *meta semantically relevant* and their *semantic proximity* can be defined as follows:

semPro($\{O_i\}$, O_2) = <context, M, (D_1, D_2), (S_1, S_2)>
where context = glb(C_{def}(DB1), C_{def}(DB2))
and M is a total 1-1 mapping between $\{O_i\}$ and S_2.

5.5.2 Attribute Entity Conflict

This conflict arises when the same entity is being modeled as an attribute in one database and a relation in another database. This kind of conflict is different from the conflicts defined in the previous and next subsections because it depends on the *database schema* and not on the *database state*. This conflict can also be considered as a part of the entity definition incompatibility (Section 5.2). Referring to the example described in the beginning of this section the attribute *stk1, stk2* in the database *DB2* correspond to relations of the same name in the database *DB3*.

Objects O_1 and O_2 can be considered *semantically relevant* as 1-1 value mappings can be established between the domains of the attribute (O_1) and the domain of the identifying attribute of the entity (O_2). It should be noted that O_1 is an attribute (property) and O_2 is an entity (class) and their definition contexts are needed to determine the identifying attribute of the entity (O_2).

semPro(O_1, O_2) = <context, total 1-1 value mapping, (D_1, D_2), _>
where context = glb(C_{def}(DB2), C_{def}(DB3))
and D_1 = Domain(O_1) and D_2 = Domain(Identifier(O_2)).

5.5.3 Data Value Entity Conflict

This conflict arises when the value of an attribute in one database corresponds to a relation in another database. Thus this kind of conflict depends on the *database state*. Referring to the example described in the beginning of this section, the values of the attribute *stkCode* in the database *DB1* correspond to the relations *stk1, stk2* in the database *DB3*.

The mappings here are established between set of entities ($\{O_i\}$) and values in the extension of an attribute (O_2). This is possible, however only *wrt* the contexts of the databases they are in. Thus the two objects may be considered to be *meta semantically relevant* and their *semantic proximity* can be defined as follows:

semPro($\{O_i\}$, O_2) = <context, M, (D_1, D_2), (S_1, S_2)>
where context = glb(C_{def}(DB1), C_{def}(DB2))
and M is a total 1-1 mapping between $\{O_i\}$ and S_2.

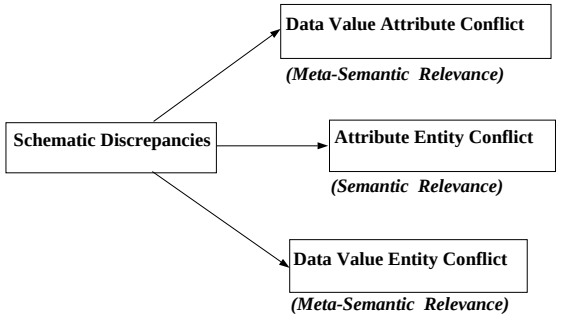

Figure 11: Schematic Discrepancies and the likely types of semantic proximities

6 Structural Similarity: A component of Semantic Similarity

In this section we propose a uniform formalism called **schema correspondences** for representation of structural similarities between objects. These are associations between objects and types defined in the various databases and can be expressed using operations from a modified object algebra. The schema correspondences so defined are a part of the semantic proximity between the two objects or types and are dependent on the context in which the semantic proximity is defined. **Projection rules** which define the relationship between schema correspondences and semantic proximity are also discussed.

6.1 Schema Correspondences: A uniform formalism for representation of Abstraction

We propose a uniform formalism to represent the mappings which are generated to represent the structural similarities between objects having schematic conflicts and some semantic affinity. This formalism is a generalization of the concept of *connectors* used to augment the relational model in [CRE87].

Given two objects O_1 and O_2, the *schema correspondence* between them can be represented as:

schCor(O_1,O_2) = <O_1,attr(O_1),O_2,attr(O_2),M>

- O_1 and O_2 are objects in the model world. They are representations or intensional definitions in the model world. They may correspond to class definitions or type definitions in a database.
- The objects enumerated above may model information at any level of representation (such as the entity or the attribute level). If an object O_i models information at the entity level,

then attr(O_i) denotes the representation of the attributes of O_i. If O_i models objects at the attribute level, then attr(O_i) is an empty set.

- M is a mapping (possibly higher-order) expressing the correspondences between objects, their attributes and the values of the objects/attributes. We use object algebra operations enumerated below.

A brief introduction to a limited object algebra

Objects are considered as collections of instances which are homogeneous and have the same type as the abstract data type associated with the object. We list a limited set of operations to manipulate objects in a database; these are very similar to those in object-oriented database literature (e.g., [SZ90])[6].

OSelect(p,O) This operation selects a set of instances of an object O satisfying a selection predicate, p.
OSelect(p,O) = $\{o|\ o \in O \wedge p(o)\}$

makeObject(C,S) Given a contextual coordinate C and a set S (which may be either a set of concepts from an ontology, an object or a type domain), defines a new object with instances having attribute C and a value from the set S as its value.
makeObject(C,S) = $\{o|\ o.C=s \wedge s \in S\}$

OProduct(O_1,O_2) Given two objects O_1 and O_2, a new object is created which has the attributes of both O_1 and O_2 and for every tuple of values in O_1 has all the tuples of values in O_2 associated with it.
$$\text{OProduct}(O_1,O_2) = \{o|\ (o.A_i=o_1.A_i \wedge A_i \in \text{attr}(O_1) \wedge o_1 \in O_1) \vee (o.A_j=o_2.A_j \wedge A_j \in \text{attr}(O_2) \wedge o_2 \in O_2)\}$$

OJoin(p,O_1,O_2) This can be thought of as a special case of the operation OProduct, except that the instances should satisfy the predicate p.
OJoin(p,O_1,O_2) = $\{o|\ o \in \text{OProduct}(O_1,O_2) \wedge p(o)\}$

Schema Correspondences and Context

Each information system exports federation objects O_F corresponding to the objects O it manages. The objects O_F are obtained by applying the constraints in the definition context $C_{def}(O)$ to the object O. The user at the federation level sees only the federation objects. The contextual coordinates C_i of the $C_{def}(O)$ act as the attributes of O_F. The exported objects O_F are associated with the objects and types defined in the database. This association might be implemented in different ways by various component systems. We use schema correspondences to express these associations. This is illustrated in Figure 12

schCor(O_F,O) = <O_F,$\{C_i|\ C_i \in C_{def}(O)\}$,O,attr(O),M>

- O_F is the exported federation object of an object O or type T defined in the database.
- The attributes of the object O_F are the contextual coordinates of the definition context $C_{def}(O)$.
- The mapping operation **map$_O$(C_i,A_i)** stores the association between contextual coordinate C_i and attribute A_i of object O whenever there exists one.
- The mapping M between O_F and O can be evaluated using the projection rules enumerated and illustrated in Section 6.2.

[6] When defining and using these operations, performance issues are ignored in favor of simplicity of description.

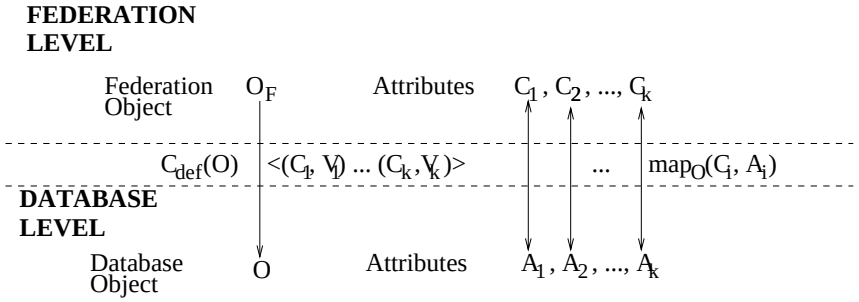

Figure 12: Schema Correspondences: Association between federation and database objects

6.2 Schema Correspondences: Projection of semPro wrt Context

We discussed in Section 3.1 how representing structural similarities is not enough to capture semantic similarity between two objects. However, for any meaningful operation to be performed on the computer, the semPro descriptor between two objects has to be mapped to a mathematical expression which would essentially express the structural correspondence between two objects. Our approach consists of the following three aspects:

The Semantic aspect: The semPro descriptor captures the real world semantics of the data in the database through context and includes intensional descriptions of:

- Objects and their attributes.
- The relationships between various objects.
- The implicit assumptions in the design of the objects.
- The constraints which the objects and attributes obey.

The federation objects are objects obtained by applying the constraints in the intensional descriptions to the database objects.

The Data Organization aspect: This refers to the actual organization of the data in the databases, e.g., the tables and views in a relational database, or the class hierarchy in object-oriented databases.

The Mapping/Abstraction aspect: The schCor descriptor as defined earlier captures the association between the federation objects and the database objects. The association uses object algebraic operations to express correspondences between the federation and the database objects. The evaluation of these associations results in the retrieval of database objects which satisfy the constraints specified in the context.

The mapping aspect can be succinctly expressed as:

$$\mathbf{schCor(O_F,O)} = \Pi_{Context}\mathbf{(semPro(O_F,O))}$$

In the rest of this section, we explain the mapping aspect with the help of examples. We first define the terminology, operations and the projection rules used to specify the semantics of the associations between the federation and database objects followed by examples illustrating them.

6.2.1 Relevant Terminology and Projection Rules

We first enumerate the operations used to specify the associations between the exported federation objects and the database objects. We shall use Cntxt, $Cntxt_1$, ... to refer to contexts and C, C_1, ... to refer to contextual coordinates. O_1, O_2, ... shall be used to refer to actual database objects whereas O_{1F}, O_{2F}, ... shall be used to denote their counterparts exported to the federation. O', O", ... shall be used to denote temporary objects to illustrate each step.

The operations are as follows:

map$_O$(C,A) The mapping operation which stores the association between the attribute C of the exported federation object O_F (which is essentially a contextual coordinate of the definition context $C_{def}(O)$ chosen from a domain specific ontology) and the attribute A of the object O.

semConstrain(<(C_i,V_i)>,semPro(O',O)) The exported federation object O_F is obtained by iteratively applying the constraints in $C_{def}(O)$ to the database object O. The *semConstrain* operation models one iteration, i.e., the application of one constraint in $C_{def}(O)$ to the database object O. Let:
- semPro(O_F,O) be defined *wrt* to $C_{def}(O)$
- C_i be a contextual coordinate of $C_{def}(O)$
- $C_{def}(O)$ = glb(<(C_i,V_i)>,Cntxt) (discussed in Section 3.3)
- semPro(O',O) be defined *wrt* Cntxt and
- O' be a temporary object obtained by applying all the constraints in Cntxt on O

Then the federation object O_F may be iteratively defined as:
semPro(O_F,O) = semConstrain(<(C_i,V_i)>,semPro(O',O))

strConstrain(map$_O$(C_i,A_i),S_i,schCor(O',O)) *strConstrain* is the structural counterpart of *semConstrain*. It maps the attributes of the federation object to the attributes of the database object. It also recomputes the mappings associated with schCor(O',O). This is done by adding a selection condition to the original mapping as follows:
O_F = OSelect((A_i $\in S_i$),O')
where there exists a mapping between O' and O from schCor(O',O)

semCondition(Cntxt,semPro(O_F,O)) In some cases, a database object O may be associated with another database object *wrt* the context *Cntxt*. The *semCondition* operation modifies the semantic proximity descriptor by *lifting* [Guh91] it into a context *(Cntxt)* different from the one *(C_{def}(O))* in which it is defined in. This operation can be defined iteratively using the semConstrain operation.
Let Cntxt = glb(<(C_i,V_i)>,$Cntxt_1$)
semCondition(Cntxt,semPro(O_F,O))
= semConstrain(<(C_i,V_i)>,semCondition($Cntxt_1$,semPro(O_F,O)))

semCombine(C_i,semPro(O',O),semPro(O",O_i)) In some cases, the definition context of an object O makes explicit an association between the database objects O and O_i. This association is typically *wrt* the association context between two objects denoted as C_{ass}(O_i,O). The *semCombine* operation models the *correlation* of information from objects O and O_i which is then exported as a part of the federation object O_F. Let:
- semPro(O_F,O) be defined *wrt* to $C_{def}(O)$
- $C_{def}(O)$ = glb(<(C_i,$O_i \circ C_{ass}(O_i,O)$)>,Cntxt)
- semPro(O',O) is defined *wrt* Cntxt
- O' be a temporary object obtained by applying the constraints in Cntxt to O
- O" be a temporary object obtained by applying the constraints in C_{ass}(O_i,O) to O_{iF}

Then the semPro(O_F,O) can be defined as:
semConstrain(<(C_i,$O_i \circ C_{ass}(O_i,O)$)>,semPro(O',O))
= semCombine(C_i,semPro(O',O),semPro(O",O_i))
where semPro(O",O_i) = semCondition(C_{ass}(O_i,O),semPro(O_{iF},O_i))

strCombine({map$_O$(C_i,A_i),map$_{O_i}$(C_i,A'$_i$)},schCor(O',O),schCor(O",O_i)) *strCombine* is the structural counterpart of semCombine. It maps the contextual coordinate C_i to the attributes of the database objects O and O_i. It correlates instances of the two objects.

This results in a join condition used to combine mappings associated with schCor(O',O) and schCor(O",O_i).

O_F = OJoin((A$_i$=A'$_i$),O',O") where there exist mappings
between O' and O from schCor(O',O) and between O" and O_i from schCor(O",O_i)

Projection Rules

We describe here a set of *projection rules* which specify the semantics of the projection operation discussed earlier in this section. The rules specify an algebra based on the operations discussed above. Here we describe them with the perspective of the role they play in mapping the federation objects to the various database objects. A detailed specification of these rules is presented in the Appendix A.1.

Rule 1: When the definition context of a database object is empty, i.e., there are no constraints which the object should satisfy, it is exported to the federation as it is without any modifications. This situation is captured by the *Empty Context Rule*.

Rule 2: The *Simple Sets Rule* deals with the case when the definition context has simple sets of values associated with each contextual coordinate. Each contextual coordinate is also associated with an attribute of a database object. The effect of this rule can be achieved with repeated applications of *Rule 3.1* but it is used to simplify the evaluation of the projection operation. An example of application of this rule is illustrated in Section 6.2.2.

Rule 3: The exported federation object O_F is obtained by iteratively applying the constraints in the definition context to the database object O. The *Simple Set Constraint Rule* deals with the case where the constraints in the context are applied iteratively to the database objects. The termination condition of the iteration is the case when the context is empty and is covered by the *Empty Context Rule*. This rule deals with the case where the constraint to be applied is of the form $\mathsf{C} \in \mathsf{S}$, where C is a contextual coordinate and S is a simple set of symbols from the ontology. This rule may also be used to apply an arbitrary constraint on a federation object.

Rule 3.1: This rule deals with the case where the contextual coordinate in the constraint is not present in the definition context in which the semPro is defined and there exists an attribute of a database object corresponding to that contextual coordinate.

Rule 3.2: This rule deals with the case where the contextual coordinate in the constraint is already present in the definition context and there exists an attribute of a database object corresponding to that contextual coordinate.

Rule 3.3: This rule deals with the case where the contextual coordinate in the constraint is not present in the definition context and there does not exist an attribute of a database object corresponding to that contextual coordinate. An example of application of this rule is illustrated in Section 6.2.3.

Rule 3.4: This rule deals with the case where the contextual coordinate in the constraint is present in the definition context and there does not exist an attribute of a database object corresponding to that contextual coordinate.

Rule 4: In some cases, a database object O_i may be associated with another database object O *wrt* an association context. The *Context Conditioning Rule* deals with the case where semPro(O_{iF},O_i) is conditioned *wrt* the association context. This involves applying the constraints in the association context to the federation object O_{iF}.

Rule 4.1: The *Empty Context Conditioning Rule* states that when the association context used to condition the semantic proximity is empty, then the semantic proximity is evaluated *wrt* the definition context. This means that the federation object O_{iF} is returned as it is without modification.

Rule 4.2: The *Constraint Conditioning Rule* deals with the case when the constraints in the association context are applied to the federation object O_{iF} iteratively. The termination condition of this iteration is when the association context is empty and is covered by the *Empty Context Conditioning Rule.*

Rule 4.3: The *Context Conditioning and semCombine Rule* deals with the case when the semantic proximity descriptor is a combination of two semantic proximities combined using the *semCombine* operation. The semantics of the *semCombine Rule* are given by Rule 5.

Rule 5: In some cases, the definition context of an object O makes explicit an association between the database objects O and O_i. This association is typically *wrt* the association context between two objects denoted as $C_{ass}(O_i,O)$. The *semCombine Rule* deals with this case and results in the generation and combination of two semantic proximities. An example of application of this rule is illustrated in Section 6.2.4.

Rule 5.1: This rule maps the contextual coordinate to the attributes in the different objects and performs the correlation of the instances of the two objects. The two attributes may either satisfy the equality predicate or any other well-defined predicate.

Rule 5.2: The *Coordinate Composition Rule* deals with the special case where the contextual coordinate in the constraint may be a composition of two contextual coordinates. Each of the contextual coordinate parts may or may not be mapped into attributes of database objects. An example of application of this rule is presented in Section 6.2.5.

6.2.2 Using ontology for an intensional description of data

In Section 3.2, we choose the contextual coordinates C_is and their values V_is from an ontology. We illustrate with the help of an example how concepts in an ontology may be mapped to the actual data in the database. Thus, the user at the federation level can view the information in the database with the help of concepts from a domain specific ontology without being aware of the underlying format of the data.

Example:

Consider an object EMPLOYEE defined in a database as follows:
EMPLOYEE(SS#,Name,Dept,SalaryType,Affiliation)

The definition context of the object EMPLOYEE may be defined as:
C_{def}(EMPLOYEE) = <(employer,[**Deptypes**∪{restypes}])
(affiliation,{teaching,research,non-teaching})(reimbursement,{salary,honorarium})>

- **Deptypes** is a type defined in the database.
- The symbols for the contextual coordinates *employer, affiliation* and *reimbursement* are taken from the ontology. The association with the attributes of EMPLOYEE is stored by the $map_{EMPLOYEE}$(C, A) operation.
- The symbols *restypes, teaching, research, non-teaching, salary* and *honorarium* may either be taken from the ontology or submitted for inclusion into the ontology by the database administrator.

As discussed in Section 6.1, we associate with definition context an object $EMPLOYEE_F$ which is exported to the federation of databases.

semPro($EMPLOYEE_F$,EMPLOYEE) = <C_{def}(EMPLOYEE),M,(dom($EMPLOYEE_F$),dom(EMPLOYEE)),_>

where M is a mapping between the domains of the two objects. The mapping relates information in the ontology to data in the database. The projection is illustrated in Figure 13.

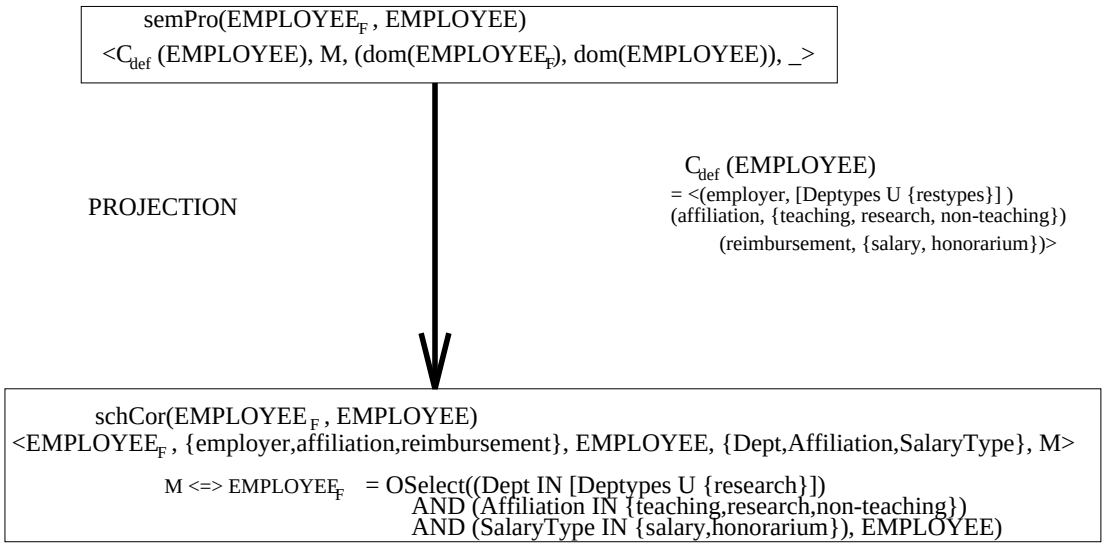

Figure 13: Mapping EMPLOYEE$_F$ to object EMPLOYEE in the database

Simple Sets Rule $\Rightarrow$

$\Pi_{C_{def}(EMPLOYEE)}$(semPro(EMPLOYEE$_F$,EMPLOYEE))
= schCor(EMPLOYEE$_F$,EMPLOYEE)
= <EMPLOYEE$_F$,{employer,affiliation,reimbursement},EMPLOYEE,
{map$_{EMPLOYEE}$(employer,Dept),map$_{EMPLOYEE}$(affiliation,Affiliation),
map$_{EMPLOYEE}$(reimbursement,SalaryType)},M>
M $\equiv$ EMPLOYEE$_F$ = OSelect(p,EMPLOYEE)
p $\equiv$ (Dept$\in$[**Deptypes**$\cup${restypes}])$\wedge$(Affiliation$\in${teaching,research,non-teaching})$\wedge$
(SalaryType$\in${salary,honorarium})

6.2.3 Domain Augmentation: Representing Extra Information

In this section, we demonstrate an interesting case where *extra information* can be stored with the intensional descriptions of objects. This extra information is represented as a constraint at the federation level. Consider the constraint All publications have research areas that are associated with Departments. This may be used to make inferences about database content, without actually accessing the database. Consider a query that asks for all publications in a research area not associated with a department. The answer to the query is an empty set which can be determined *without* actually accessing the database.

The constraint involving research areas can be represented in C_{def}(PUBLICATION) and expressed using the contextual coordinate *researchArea*. However the information about the research areas of a publication is not modeled by the existing database object PUBLICATION(Id,Title,Journal).

The definition context of the object PUBLICATION is defined as:
C_{def}(PUBLICATION) = <(researchArea,**Deptypes**)> where **Deptypes** is a type defined in the database.

The query discussed above can be processed without accessing the database if the constraint involving research areas is part of the exported federation object. Because the contextual coordinate *researchArea* is not modeled in the database, the projection algorithm creates a new object corresponding to the research areas by using the *makeObject* operation. This new object is then associated with the database object PUBLICATION by using the *OProduct* operation. The above results in the augmentation of the domain of the database object PUBLICATION

and is expressed in Appendix A.1 *(Rule 3.3)*.
$\text{dom}(\text{PUBLICATION}_F) \subseteq \text{dom}(\text{Id}) \times \text{dom}(\text{Title}) \times \text{dom}(\text{Journal}) \times \textbf{Deptypes}$.

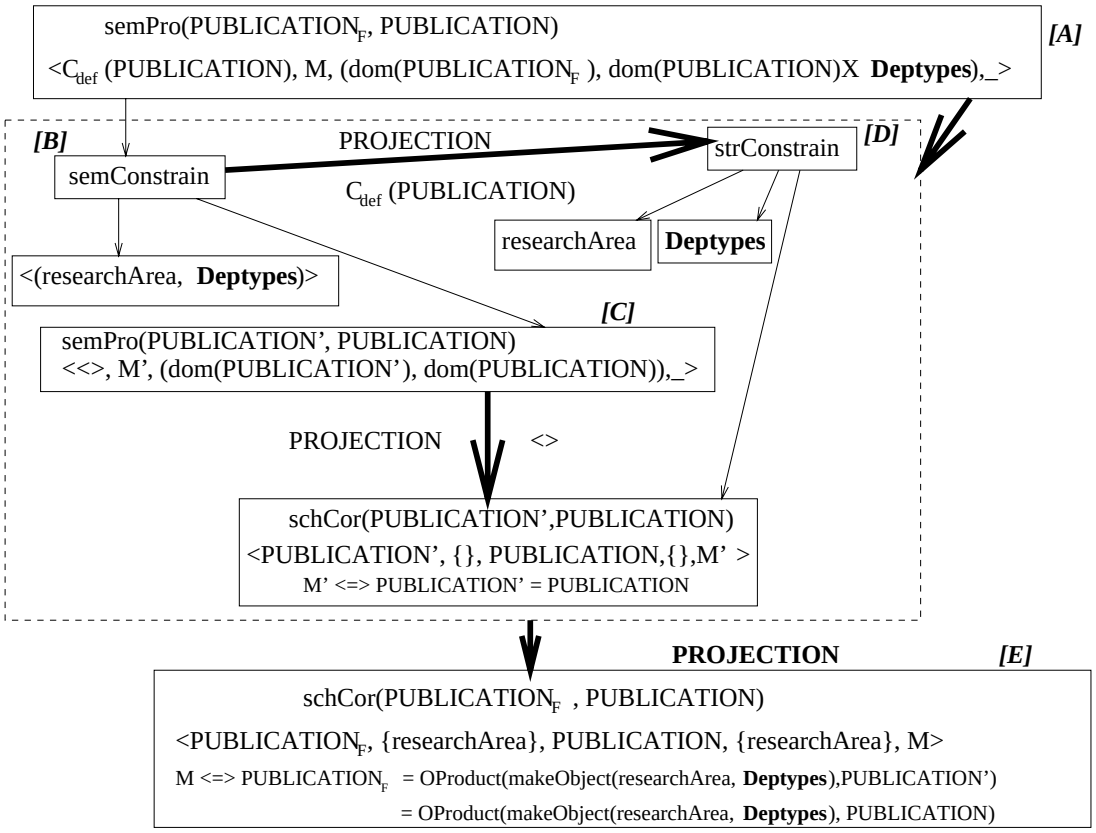

Figure 14: Domain Augmentation: Mapping PUBLICATION_F to object PUBLICATION in the database

The projection operation is diagrammatically illustrated in Figure 14.

[**A**] semPro(PUBLICATION$_F$,PUBLICATION) is evaluated *wrt* C_{def}(PUBLICATION). The definition context expresses extra information about the object PUBLICATION not modeled in the database. This step illustrates the augmentation of dom(PUBLICATION). Let:
- C_{def}(PUBLICATION) = glb(<(researchArea, Deptypes)>, <>)
- semPro(PUBLICATION',PUBLICATION) be defined *wrt* <>
- PUBLICATION' be a temporary object

The Domain augmentation takes place as follows:
Simple Set Constraint Rule (New Constraint, Non-existing attribute) ⇒ (Step [**B**])

semPro(PUBLICATION$_F$,PUBLICATION)
= semConstrain(<(researchArea,**Deptypes**)>, semPro(PUBLICATION',PUBLICATION))

- Let M' be the mapping between PUBLICATION' and PUBLICATION returned by Step [**C**].
- The constraint about research areas is incorporated in the exported federation object PUBLICATION_F by using the mapping M. The evaluation of the mapping is illustrated in Steps [**D**,**E**].
- The resulting augmentation of the domain of the object PUBLICATION is reflected in the definition of the modified semPro descriptor:

semPro(PUBLICATION$_F$,PUBLICATION)
= <C_{def}(PUBLICATION),M,(dom(PUBLICATION$_F$),**dom(PUBLICATION)**× **Deptypes**),_>

[**C**] Empty Context Rule ⇒
M' ≡ PUBLICATION'=PUBLICATION

[**D**,**E**] Simple Set Constraint Rule (Rule 3.3) ⇒

schCor(PUBLICATION$_F$,PUBLICATION)
= $\Pi_{C_{def}(PUBLICATION)}$(semConstrain(<(researchArea,**Deptypes**)>,
semPro(PUBLICATION',PUBLICATION)))
= strConstrain({researchArea},**Deptypes**,schCor(PUBLICATION',PUBLICATION))
M ≡ PUBLICATION$_F$=OProduct(makeObject(researchArea,**Deptypes**),PUBLICATION')
=OProduct(makeObject(researchArea,**Deptypes**),PUBLICATION)

6.2.4 Representing relationships between objects

In this section, we illustrate with the help of an example how context can be used to capture relationships between objects which may not be represented in the database. We illustrate a case where the definition context of the object HAS-PUBLICATION captures its relationships with another database object EMPLOYEE in an intensional manner. These relationships are *not stored* in the database and the evaluation of the semPro descriptor results in *extra information* being associated with the federation object HAS-PUBLICATION$_F$. A naive user will ordinarily not be aware of this relationship.

Example:

Consider objects EMPLOYEE and PUBLICATION defined earlier and an object in the same database which represents a relationship between employees and the publications they write,
HAS-PUBLICATION(SS#,Id)
C_{def}(HAS-PUBLICATION) = <(author,EMPLOYEE∘ <(affiliation,{research})>)>

This evaluation of the semPro descriptor has been diagrammatically illustrated in Figure 15.

[**A**] semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION) is evaluated *wrt* C_{def}(HAS-PUBLICATION).
The definition context makes explicit the relationship between HAS-PUBLICATION and EMPLOYEE. This step illustrates how the correlation of the instances of EMPLOYEE and HAS-PUBLICATION is done to satisfy the constraints in the definition context. Let:
- C_{def}(HAS-PUBLICATION) = glb(<(author,EMPLOYEE∘C_{ass}(EMPLOYEE,HAS-PUBLICATION))>,<>)
- semPro(HAS-PUBLICATION',HAS-PUBLICATION) be defined *wrt* <>
- C_{ass}(EMPLOYEE,HAS-PUBLICATION) = <(affiliation,{research})>
- HAS-PUBLICATION' be a temporary object
- EMPLOYEE' be a temporary object obtained by applying the constraints in C_{ass}(EMPLOYEE,HAS-PUBLICATION) to EMPLOYEE$_F$

semCombine Rule ⇒

semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION)
= semCombine(author,semPro(HAS-PUBLICATION',HAS-PUBLICATION),
semPro(EMPLOYEE',EMPLOYEE))

- Let M' be the mapping between HAS-PUBLICATION' and HAS-PUBLICATION returned by Step [**B**].
- semPro(EMPLOYEE',EMPLOYEE)
= semCondition(C_{ass}(EMPLOYEE,HAS-PUBLICATION),semPro(EMPLOYEE$_F$,EMPLOYEE))
Let M" be the mapping between EMPLOYEE' and EMPLOYEE returned by Step [**C**].
- map$_{EMPLOYEE}$(author,SS#) and map$_{HAS-PUBLICATION}$(author,SS#)

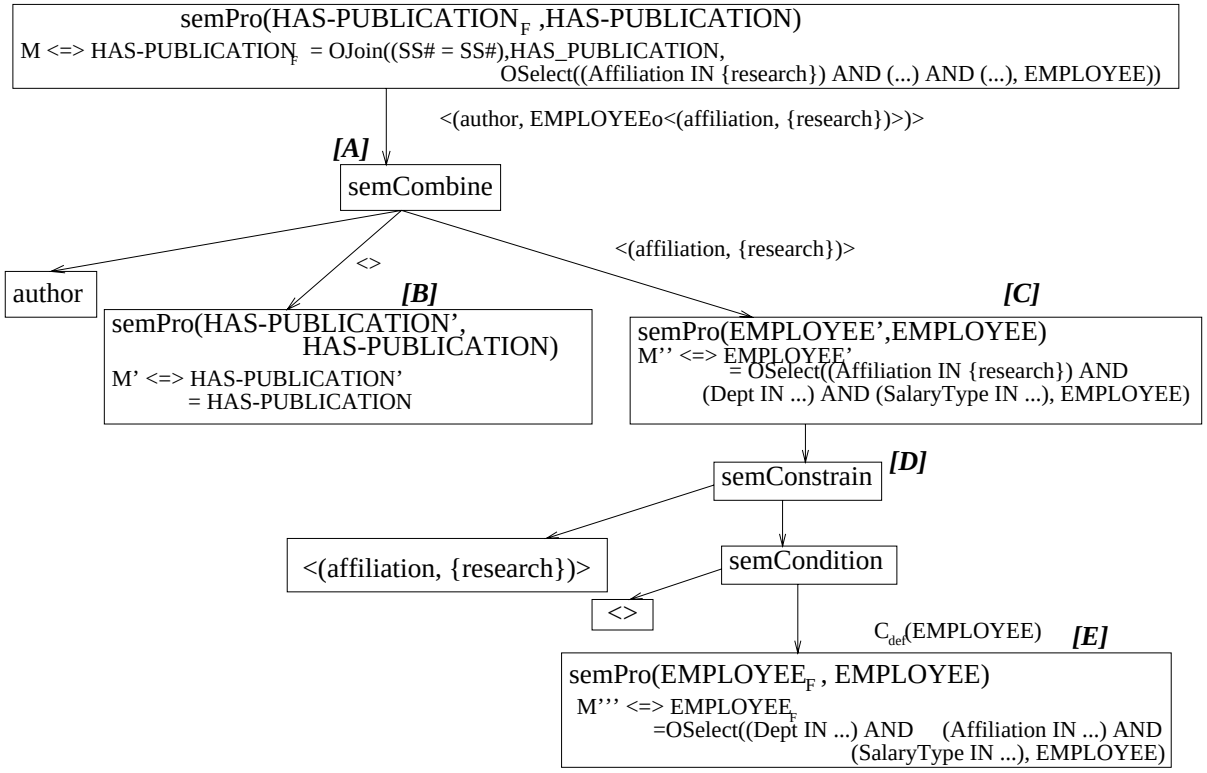

Figure 15: Correlation of Information between HAS-PUBLICATION and EMPLOYEE

Rule 5.1 ⇒

$M \equiv$ HAS-PUBLICATION$_F$ = OJoin((SS#=SS#),HAS-PUBLICATION',EMPLOYEE')
= OJoin((SS#=SS#),HAS-PUBLICATION,EMPLOYEE')
.... M' From Step [**B**]
= OJoin((SS#=SS#),HAS-PUBLICATION,
OSelect((Affiliation$\in${research})$\wedge$(...)$\wedge$(...),EMPLOYEE))
.... M" From Step [**C**]

[**B**] Empty Context Rule ⇒
M' $\equiv$ HAS-PUBLICATION'=HAS-PUBLICATION

[**C**] In this step we show how the constraints in the association context are applied to the federation object EMPLOYEE$_F$. This is done *before* correlation of the instances of EMPLOYEE and HAS-PUBLICATION as only employees who are researchers have publications.
C_{ass}(EMPLOYEE,HAS-PUBLICATION)=glb(<(affiliation,{research})>, <>)
Constraint Conditioning Rule ⇒

semCondition(C_{ass}(EMPLOYEE,HAS-PUBLICATION),
semPro(EMPLOYEE$_F$,EMPLOYEE))
= semConstrain(<(affiliation,{research})>,
semCondition(<>,semPro(EMPLOYEE$_F$,EMPLOYEE)))
.... Illustrated in Step [**D**]
= semConstrain(<(affiliation,{research})>,semPro(EMPLOYEE$_F$,EMPLOYEE))
.... Empty Context Conditioning Rule

Let M'' be the mapping returned by Step [**E**] between EMPLOYEE$_F$ and EMPLOYEE.

Rule 3.2 ⇒

M" ≡ EMPLOYEE'=OSelect((Affiliation∈{research}),EMPLOYEE$_F$)
= OSelect((Affiliation∈{research}),
OSelect((Affiliation∈{research,teaching,non-teaching})∧(...)∧(...),EMPLOYEE)
.... M"' From Step **[E]**
= OSelect((Affiliation∈{research})∧(...)∧(...),EMPLOYEE)

[E] This step illustrates the association between the federation object EMPLOYEE$_F$ and the database object EMPLOYEE and has been discussed in detail in Section 6.2.2. The association is given by:

M"' ≡ EMPLOYEE$_F$
=OSelect((Affiliation∈{research,teaching,non-teaching})∧(...)∧(...)),EMPLOYEE)

6.2.5 Composition of Contextual Coordinates: Representing extra information

In this section, we illustrate an example in which the information that the contextual coordinate *researchInfo* is a composition of two contextual coordinates (*researchArea* and *journalTitle*) is obtained from the ontology of the domain. This is then used to correlate information between the objects PUBLICATION and JOURNAL. However, the contextual coordinate researchArea has not been modeled for the object PUBLICATION. Thus, this results in *extra information* about the relevant journals and research areas being associated with the object PUBLICATION, even though no information about research areas is modeled for PUBLICATION.

Example:

Consider a database containing the following objects:
PUBLICATION(Id, Title, Journal)
C_{def}(PUBLICATION)
= <(researchInfo,JOURNAL∘ <(researchArea,Deptypes)(journalTitle,JournalTypes)>)>
JOURNAL(Title, Area) C_{def}(JOURNAL) = <>

The *correlation of information* is illustrated diagrammatically in Figure 16.

[A] semPro(PUBLICATION$_F$,PUBLICATION) is evaluated *wrt* C_{def}(PUBLICATION)
The definition context makes explicit the relationship between PUBLICATION and JOURNAL. This step illustrates the generation of the two semPro descriptors, one for applying the remaining constraints in C_{def}(PUBLICATION) to PUBLICATION and the other for applying the constraints in C_{ass}(JOURNAL,PUBLICATION) to JOURNAL$_F$. Let:
- C_{def}(PUBLICATION)
  = glb(<(researchInfo,JOURNAL∘ <(researchArea,Deptypes)(journalTitle,JournalTypes)>)>,<>)
- semPro(PUBLICATION', PUBLICATION) be defined *wrt* <>
- C_{ass}(JOURNAL,PUBLICATION) = <(researchArea,Deptypes)(journalTitle,JournalTypes)>
- PUBLICATION' be a temporary object
- JOURNAL' be a temporary object obtained by applying the constraints in
  C_{ass}(JOURNAL,PUBLICATION) to JOURNAL

semCombine Rule ⇒

semPro(PUBLICATION$_F$,PUBLICATION)
= semCombine(researchInfo,semPro(PUBLICATION',PUBLICATION),
semPro(JOURNAL',JOURNAL))

- Let M' be the mapping between PUBLICATION' and PUBLICATION returned by Step **[B]**.
- semPro(JOURNAL',JOURNAL)
  = semCondition(C_{ass}(JOURNAL,PUBLICATION),semPro(JOURNAL$_F$,JOURNAL))
  Let M" be the mapping between JOURNAL' and JOURNAL returned by Step **[C]**.

[B] Empty Context Rule ⇒
The mapping M' associated with schCor(PUBLICATION',PUBLICATION) is:
M' ≡ PUBLICATION'=PUBLICATION

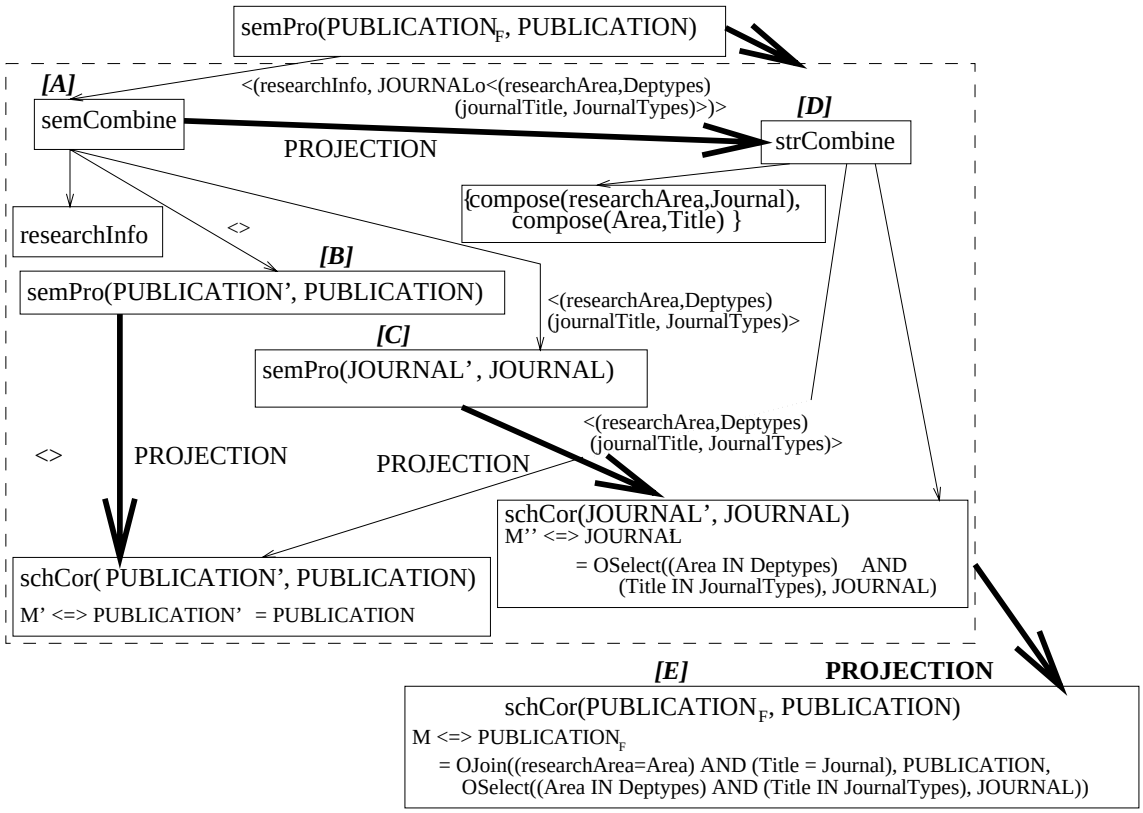

Figure 16: Correlation between PUBLICATION and JOURNAL due to composition of contextual coordinates

[**C**] C_{ass}(JOURNAL,PUBLICATION)
= glb(<(researchArea,Deptypes)>,glb(<(journalTitle,JournalTypes)>,<>))
2 applications of Constraint Conditioning Rule and 1 application of Empty Context Conditioning Rule ⇒

semCondition(C_{ass}(JOURNAL,PUBLICATION),semPro(JOURNAL$_F$,JOURNAL))
= semConstrain(<(researchArea,Deptypes)>,
semConstrain(<(journalTitle,JournalTypes)>,semPro(JOURNAL$_F$,JOURNAL)))

2 applications of Rule 3.2 and C_{def}(JOURNAL) = <> ⇒
The mapping M" associated with schCor(JOURNAL',JOURNAL) is:
M" ≡ JOURNAL' = OSelect((Area∈Deptypes)∧(Title∈JournalTypes),JOURNAL)

[**D**] semPro(PUBLICATION$_F$,PUBLICATION) is evaluated by applying the *Coordinate Composition Rule*. The final result is illustrated in Step [**E**]. This step illustrates how information about the research areas of the publications is propagated to PUBLICATION even though there is no information about research areas stored in the object PUBLICATION. This is achieved by the composition of contextual coordinates obtained from the domain ontology.

- researchInfo = compose(researchArea,journalTitle)
  Coordinate Composition Rule ⇒

map$_{PUBLICATION}$(researchInfo,X)
= compose(map$_{PUBLICATION}$(researchArea,NA),map$_{PUBLICATION}$(journalTitle,Journal))
map$_{JOURNAL}$(researchInfo,Y)
= compose(map$_{JOURNAL}$(researchArea,Area),map$_{JOURNAL}$(journalTitle,Title))

- The mapping M associated with schCor(PUBLICATION$_F$,PUBLICATION) is given by:

strCombine({map$_{PUBLICATION}$(researchInfo,X),map$_{JOURNAL}$(researchInfo,Y)},
schCor(PUBLICATION',PUBLICATION),schCor(JOURNAL',JOURNAL))
M ≡ PUBLICATION$_F$ = OJoin((X=Y),PUBLICATION',JOURNAL')
= OJoin((researchArea=Area)∧(Title=Journal),PUBLICATION',JOURNAL')
= OJoin((researchArea=Area)∧(Title=Journal),PUBLICATION,JOURNAL')
.... mapping M' from Step **[B]**
= OJoin((researchArea=Area)∧(Title=Journal),PUBLICATION,
OSelect((Area∈Deptypes)∧(Title∈JournalTypes),JOURNAL))
.... mapping M'' from Step **[C]**

The constraint researchArea ∈ Deptypes propagates to PUBLICATION. This is because when the correlation takes place between JOURNAL and PUBLICATION (refer to Step **[E]**):

- Only journals belong to the research areas corresponding to the departments are selected (OSelect((Area IN Deptypes) AND ... ,JOURNAL)).
- The join condition (Title = Journal) ensured that only those articles which are from the research areas corresponding to the departments are exported to the federation (OJoin((researchArea=Area) AND (Title = Journal), ...)).
- This is achieved in-spite of the attribute Area not being modeled for PUBLICATION. Thus there is a selective and implicit domain augmentation of Deptypes to PUBLICATION through the join condition.

6.2.6 Representation of Incomplete Information

The intensional description of the definition contexts can be easily used to represent incomplete information. Traditional database approaches have used NULL values to represent incomplete information. The semantics of NULL values is not always clear (e.g., a NULL value can mean unknown or not applicable) and this can be a problem while retrieving incomplete information from the database. We can use intensional descriptions in an attempt to describe incomplete information and to avoid the problems associated with NULL values.

Example: Consider the following definition context of the object PUBLICATION.

C_{def}(PUBLICATION) = <(title,{x|substring(x)="abortion"})>

This represents a constraint on the instances of the object PUBLICATION such that all the titles should have the word "abortion" in them. This does not specify the title of each instance of PUBLICATION completely. This information can be represented with the object PUBLICATION$_F$ at the federation level and can help in querying the database in face of incomplete information.

6.3 Applications of Context

In Section 6.2, we defined and illustrated with examples the relationship between schema correspondences and semantic proximity. We have defined *projection rules* which define schema correspondences as the projection of the semPro descriptor *wrt* the context. Earlier work on mapping intensional descriptions of concepts to SQL queries on relational databases has been reported in [BB93]. In our approach however, the mappings expressed using object algebra operations are also associated with the intensional contextual descriptions. Whenever the context changes, we also keep track of the associated changes in the schema correspondences. Rules modeling the changes in the schema correspondences (and hence the mappings) due to changes in context are presented in [KS95b].

We look at examples in which the semPro descriptors are *lifted* [Guh91] to different contexts. Lifting a semPro to a different context means re-evaluating the semPro in a context which is different from the one it was defined in the first place. We show in [KS95b] how query processing

can be implemented by the comparison of the definition contexts of the objects in the database with the query context. We have illustrated:

- How the modification of schema correspondences due to changes in context lead to *information focusing*.
- How changes in the definition context of one object leads to the modification of schema correspondence of a related object.
- How constraints from the query contexts can be applied to an object stored in a database. This results in modification of the schema correspondences and results in information focusing.
- How the query context can form the basis of correlation of information across different databases.

7 Related Work

A simple observation made by various researchers in the field of multidatabases, which is also the central premise of this paper is that it is essential to associate abstractions/mappings between objects with the context of comparison to capture semantic similarity. Some significant attempts are the **semantic proximity** proposal by Sheth and Kashyap [SK92], the **context building** approach by Ouksel and Naiman [ON93], the **context interchange** approach by Sciore et al. [SSR92] and the **common concepts** approach by Yu et al. [YSDK91]. We have related the above attempts to semantic proximity. A detailed discussion of these can be found in [KS95c].

There have been attempts to use an attribute-value based representation for capturing similarities in various areas of research. Larson et al [LNE89] use a set of fixed descriptors to capture similarities between attributes. Sciore et al [SSR92] use meta-attributes to represent context. In linguistics [CMG90], context has been represented using a set of context coordinates subject to certain conditions. Similar attempts have also been made for documents in text retrieval (using thematic roles) [VD92] and for clustering similar objects (using code words) in [ML92]. We have abstracted out the commonalities in these approaches in our representation of context. However, we differ from Sciore et al [SSR92] and Ouksel et al [ON93] in the following aspects:

- Sciore et al [SSR92] represent the context at the extensional level, i.e., at the level of data values and object instances. We represent context at an intensional level, i.e. at the level of the database schema. This gives us an opportunity to represent constraints about objects which cannot be captured at the extensional level. We also view the context of an object as a collection of constraints on an object which may not be represented in the database schema.
- Ouksel et al represent context as a collection of ISCAs (inter-schema correspondence assertions) which are essentially structural correspondences between schema elements in different databases. In our approach schema correspondences are associated with the context and are not considered part of the context. They are used to relate semantic information with the actual data in the database.
- The meta-attributes and their values are taken from the ontology of the application domain being modeled by the database. Issues of combining ontologies and scalability are beyond the scope of this paper but are discussed in [KS94a, KS95a].
- We have also defined operations to compare the specificity of contexts, and to manipulate and reason about them. Based on the partial order induced by the specificity relationship, we organize the contexts as a meet semi-lattice. Inferences on a new context *wrt* the knowledge present in the context set can now be supported by determining its position in the semi-lattice.

We have expressed our context descriptions in a description logic like language. We have used CLASSIC [BBMR89] expressions in this paper. Other well known description logic languages are KL-ONE [BS85], LOOM [Mac87] and BACK [vLNPS87]. The advantage of using CLASSIC is that it is sufficiently expressive and has a polynomial time classification algorithm.

Classification or taxonomies of *schematic differences* appear in [DH84, BOT86, CRE87, KLK91, KS91]. In this paper we present what we believe is a comprehensive taxonomy of schematic conflicts which subsumes most of the taxonomies found in literature (Table 3 in Appendix A.2). We refined the broad definition of domain incompatibility and entity definition incompatibility given in [CRE87]. Our classification consists of conflicts arising out of inconsistencies in the database state [BOT86], conflicts due to representation at differing levels of abstraction [DH84] and conflicts when data in one database corresponds to metadata in another [DAODT85, KLK91].

8 Conclusions and Future Work

An essential prerequisite to achieving interoperability in a multidatabase environment is to be able to identify semantically similar data in different database systems. Another key issue attracting wide attention with attempts to build a National Information Infrastructure, is the issue of querying a large number of autonomous databases without prior knowledge of their information content. It is therefore important to capture the semantic content of these databases in as explicit a manner as possible.

We discussed the inadequacy of structural similarity and how semantics cannot be captured by purely mathematical formalisms. This led us to make a case for the explicit identification and representation of context in a multidatabase environment. We define the concept of *semantic proximity*, using which we represent the degrees of semantic similarities between the objects [SK92]. The *context* of comparison of these objects is the fulcrum of the semantic proximity. We propose an explicit though partial representation of context in a multidatabase environment. We have also defined the concept of *schema correspondences*, using which we represent the structural similarities between objects.

We demonstrate the reconciliation of the dual schematic *vs* semantic perspectives. This is done by associating the mapping/abstraction between objects in different databases with the context of the semantic proximity defined between them. This association enables us to determine qualitative measures of semantic similarity such as *equivalence, relationship, relevance, resemblance and incompatibility* and develop a semantic taxonomy. We also enumerate the various schematic heterogeneities and the possible semantic similarities between them.

We have also defined the concept of *schema correspondences*, using which we represent the structural similarities between objects. Though it is known that representing structural similarities is inadequate to capture semantic similarity between two objects, for any meaningful operation to be performed on the computer, the semPro descriptor between two objects has to be mapped to a mathematical expression which would essentially express the structural correspondence between them. We have defined the schema correspondences as a projection *wrt* context of the semantic proximity between the objects.

Besides helping to reconcile the semantic and the structural perspectives, it also enables us to represent extra knowledge about the database objects. This includes domain specific constraints obtained from an ontology and implicit relationships between objects in the databases. We also demonstrate how extra information not modeled for a database object may be associated with it. This enables inferences to be drawn at the federation level without accessing the databases. Some of these inferences involve extra knowledge and would not have been possible even if the objects in the databases were accessed.

These inferences are modeled as changes in the context and the associated schema correspondences. It enables *information focusing* as some inferences affect the schema correspondences to retrieve only the data relevant to the query. It enables *information correlation* as one can specify constraints relating different objects in the context. The computation of the resulting schema correspondences enables the correlation of the appropriate instances of the objects. These have been discussed briefly in the paper due to space constraints. The reader may refer to [KS95b] for details.

The context is the key component in capturing the semantic content of the information present in the various databases. In any attempt to represent the context of objects in a database, issues of language and vocabulary become important. We are looking into the possibility of the Knowledge Interchange Format [GF92] and description logic based languages [BS85, BBMR89, Mac87, PS84, vLNPS87, KBR86] for context representation. In designing the definition context of an object, it is necessary to choose the contextual coordinates and their values in a controlled manner. We are experimenting on using domain specific ontologies to construct these contexts in a methodical manner. In cases where a domain ontology is not readily available, research is required to enable semi-automatic generation of ontologies. We are looking at Clustering and Information Retrieval techniques for semi-automatic generation of ontologies.

A complementary problem is that of presenting the ontologies to the user in a methodical manner to enable him to construct the query contexts for retrieving information from a federation of databases. Tools to present these ontologies to users and information system designers must be developed to facilitate context design and representation.

There should be an agreement on the meaning of the terms used in the ontologies for construction of the definition contexts on one hand and those used in the ontologies for the construction of the query contexts on the other. Thus, either a common ontology is required, or the correspondence between the terms in the various ontologies needs to be established. We are looking into re-using existing ontologies and classifications to establish/maintain this agreement in a scalable manner.

References

[ACHK93] Y. Arens, C. Chee, C. Hsu, and C. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2), June 1993.

[BB93] A. Borgida and R. Brachman. Loading Data into Description Reasoners. In *Proceedings of 1993 ACM SIGMOD*, May 1993.

[BBMR89] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. Classic: A structural data model for objects. In *Proceedings of ACM SIGMOD-89*, 1989.

[BOT86] Y. Breitbart, P. Olson, and G. Thompson. Database Integration in a Distributed Heterogeneous Database System. In *Proceedings of the 2nd IEEE Conference on Data Engineering*, February 1986.

[BS85] R. Brachman and J. Scmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), February 1985.

[CHS91] C. Collet, M. Huhns, and W. Shen. Resource Integration using a Large Knowledge Base in Carnot. *IEEE Computer*, December 1991.

[CMG90] G. Chierchia and S. McConnell-Ginet. *Meaning and Grammar: An Introduction to Semantics*, chapter 6. MIT Press Cambridge MA, 1990.

[CRE87] B. Czejdo, M. Rusinkiewicz, and D. Embley. An approach to Schema Integration and Query Formulation in Federated Database Systems. In *Proceedings of the 3rd IEEE Conference on Data Engineering*, February 1987.

[DAODT85] S. Deen, R. Amin, G. Ofori-Dwumfuo, and M. Taylor. The architecture of a Generalised Distributed Database System PRECI*. *IEEE Computer*, 28(4), 1985.

[DH84] U. Dayal and H. Hwang. View definition and Generalization for Database Integration of a Multidatabase System. *IEEE Transactions on Software Engineering*, 10(6), November 1984.

[ELN86] R. Elmasri, J. Larson, and S. Navathe. Schema Integration Algorithms for Federated Databases and Logical Database Design. Technical report, Honeywell Corporate Systems Develpment Division, Golden Valley, MN, 1986.

[EN89] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummins, 1989.

[FKN91] P. Fankhauser, M. Kracker, and E. Neuhold. Semantic vs. Structural resemblance of Classes. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[GF92] M. Genesereth and R. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[Gru93] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge-Based Systems*, 5(2), June 1993.

[Guh90] R. V. Guha. Micro-theories and Contexts in Cyc Part I : Basic Issues. Technical Report ACT-CYC-129-90, Microelectronics and Computer Technology Corporation, Austin TX, June 1990.

[Guh91] R. Guha. Contexts: A Formalization and some Applications. Technical Report STAN-CS-91-1399-Thesis, Department of Computer Science, Stanford University, December 1991.

[HK87] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3), 1987.

[KBR86] T. Kaczmarek, R. Bates, and G. Robins. Recent developments in NIKL. In *Proceedings AAAI-86*, 1986.

[KLK91] R. Krishnamurthy, W. Litwin, and W. Kent. Language features for Interoperability of Databases with Schematic Discrepancies. In *Proceedings of 1991 ACM SIGMOD*, May 1991.

[KS91] W. Kim and J. Seo. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, 24(12), December 1991.

[KS93] V. Kashyap and A. Sheth. Schema Correspondences between Objects with Semantic Proximity. Technical Report DCS-TR-301, Department of Computer Science, Rutgers University, October 1993.

[KS94a] V. Kashyap and A. Sheth. Semantics-based Information Brokering. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM)*, November 1994.

[KS94b] V. Kashyap and A. Sheth. Semantics-based Information Brokering: A step towards realizingthe Infocosm. Technical Report DCS-TR-307, Department of Computer Science, Rutgers University, March 1994.

[KS95a] V. Kashyap and A. Sheth. Controlled Vocabulary Sharing for Query processing in Global Information Systems. Technical report, LSDIS Lab, University of Georgia, February 1995. http://www.cs.uga.edu/LSDIS/infoquilt.

[KS95b] V. Kashyap and A. Sheth. Semantic and Schematic Similarities between Databases Objects: A Context-based approach. Technical report, LSDIS Lab, University of Georgia, January 1995. http://www.cs.uga.edu/LSDIS/infoquilt.

[KS95c] V. Kashyap and A. Sheth. Semantic similarities between Objects in Multiple Databases. In A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors, *Heterogeneous Distributed Databases*, chapter 3. Morgan Kaufmann, 1995. (in preparation).

[LA86] W. Litwin and A. Abdellatif. Multidatabase Interoperability. *IEEE Computer*, 19(12), December 1986.

[LG90] D. Lenat and R. V. Guha. *Building Large Knowledge Based Systems : Representation and Inference in the Cyc Project.* Addison-Wesley Publishing Company Inc, 1990.

[LNE89] J. Larson, S. Navathe, and R. Elmasri. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering*, 15(4), 1989.

[Mac87] R. MacGregor. A deductive pattern matcher. In *Proceedings AAAI-87*, 1987.

[McC93] J. McCarthy. Notes on formalizing Context. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1993.

[ML92] S. H. Myaeng and M. Li. Building Term Clusters by acquiring Lexical Semantics from a Corpus. In *Proceedings of the CIKM*, 1992.

[MS95] D. McLeod and A. Si. The Design and Experimental Evaluation of an Information Discovery Mechanism for Networks of Autonomous Database Systems. In *Proceedings of the 11th IEEE Conference on Data Engineering*, February 1995.

[ON93] A. Ouksel and C. Naiman. Coordinating Context Building in Heterogeneous Information Systems. *Journal of Intelligent Information Systems*, 1993.

[PM88] J. Peckham and J. Maryanski. Semantic data models. *ACM Computing Surveys*, 20(3), 1988.

[PS84] P. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings IEEE Workshop on Principle of Knowledge-Based Systems*, 1984.

[RSK91] M. Rusinkiewicz, A. Sheth, and G. Karabatis. Specifying Interdatabase Dependencies in a Multidatabase Environment. *IEEE Computer*, 24(12), December 1991.

[SG89] A. Sheth and S. Gala. Attribute relationships : An impediment in automating Schema Integration. In *Proceedings of the NSF Workshop on Heterogeneous Databases*, December 1989.

[She91] A. Sheth. Semantic issues in Multidatabase Systems. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[Sho91] Y. Shoham. Varieties of Context, 1991.

[SK92] A. Sheth and V. Kashyap. So Far (Schematically), yet So Near (Semantically). *Invited paper in Proceedings of the IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems, DS-5*, November 1992.

[SL90] A. Sheth and J. Larson. Federated Database Systems for managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys*, 22(3), September 1990.

[SRK92] A. Sheth, M. Rusinkiewicz, and G. Karabatis. Using Polytransactions to manage Independent Data. In *Database Transaction Models*, 1992.

[SSR92] E. Sciore, M. Siegel, and A. Rosenthal. Context Interchange using Meta-Attributes. In *Proceedings of the CIKM*, 1992.

[SZ90] G. Shaw and S. Zdonik. A Query Algebra for Object-Oriented databases. In *Proceedings of the 6th IEEE Conference on Data Engineering*, February 1990.

[VD92] D. A. Voss and J. R. Driscoll. Text Retrieval using a Comprehensive Lexicon. In *Proceedings of the CIKM*, 1992.

[vLNPS87] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. Technical Report KIT Report 41, Technical University of Berlin, Berlin, F.R.G., 1987.

[Wie94] G. Wiederhold. Interoperation, Mediation and Ontologies. In *FGCS Workshop on Heterogeneous Cooperative Knowledge-Bases*, December 1994.

[YSDK91] C. Yu, W. Sun, S. Dao, and D. Keirsey. Determining relationships among attributes for Interoperability of Multidatabase Systems. In *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*, April 1991.

Appendix

A.1 Detailed Specification of Projection Rules

semPro(O_{1F},O_1) = <Cntxt,M,(dom(O_{1F}),dom(O_1)),_>

Π_{Cntxt}(semPro(O_{1F},O_1)) = schCor(O_{1F},O_1) = <O_{1F},{C_i| C_i ∈Cntxt},O_1,attr(O_1),M>

Rule 1: *Empty Context Rule*, i.e. Cntxt = <>

schCor(O_{1F},O_1) = <O_{1F},ϕ,O_1,ϕ,M> ⇒ M ≡ O_{1F}=O_1

Rule 2: *Simple Sets Rule*, i.e. Cntxt = <(C_1,S_1)...(C_k,S_k)>

schCor(O_{1F},O_1) = <O_{1F},{C_i|C_i ∈Cntxt},O_1,{A_i|map$_{O_1}$(C_i,A_i) exists},M>
M ≡ O_{1F}=OSelect(p,O_1), where p ≡ (A_1 ∈S_1) ∧...∧ (A_k ∈S_k)

Rule 3: *Simple Set Constraint Rule*, when Cntxt = glb(<(C_j,S_j)>,Cntxt$_1$)

semPro(O_{1F},O_1) = semConstrain(<(C_j,S_j)>,semPro(O',O_1))
where semPro(O',O_1) is defined *wrt* Cntxt$_1$ and
O' is a temporary object obtained by applying constraints in Cntxt$_1$ on O_1

schCor(O_{1F},O_1) = Π_{Cntxt}(semConstrain(<(C_j,S_j)>,semPro(O',O_1))
= strConstrain(map$_{O_1}$(C_j,A_j),S_j,schCor(O',O_1))
where the mapping M' associated with schCor(O',O_1) is given by:
M' ≡ O'=OSelect(p,O_1)

Rule 3.1: *New Constraint, Existing Attribute*, i.e. C_j ∉Cntxt$_1$, map$_{O_1}$(C_j,A_j) exists.
The Mapping M associated with schCor(O_{1F},O_1) is given as:

M ≡ O_{1F}=OSelect((A_j ∈S_j),O')=OSelect((A_j ∈S_j),OSelect(p,O_1))
=OSelect((A_j ∈S_j)∧p,O_1)

Rule 3.2: *Existing Constraint, Existing Attribute*, i.e. C_j ∈Cntxt$_1$, map$_{O_1}$(C_j,A_j) exists
Suppose (C_j,S'$_j$)∈Cntxt$_1$.
Then the mapping M' associated with schCor(O',O_1) may be written as:
M' ≡ O'=OSelect(p'∧(A_j ∈S'$_j$),O_1) where p ≡ p'∧(A_j ∈S_j)
The Mapping M associated with schCor(O_{1F},O_1) is then given as:

M ≡ O_{1F}=OSelect((A_j ∈S_j),O')=OSelect((A_j ∈S_j),OSelect(p'∧(A_j ∈S'$_j$),O_1))
= OSelect(p'∧(A_j ∈S_j∩S'$_j$),O_1)

Rule 3.3: *New Constraint, Non-existing attribute*, i.e. C_j ∉Cntxt$_1$, map$_{O_2}$(C_j,A_j) does not exist
The Mapping M associated with schCor(O_{1F},O_1) is given as:

M ≡ O_{1F}=OProduct(makeObject(C_j,S_j),O')
=OProduct(makeObject(C_j,S_j),OSelect(p,O_1))

Rule 3.4: *New Constraint, Non-existing Attribute*, i.e. C_j ∈Cntxt$_1$, map$_{O_2}$(C_j,A_j) does not exist
Suppose (C_j,S'$_j$)∈Cntxt$_1$.
Then the mapping M' associated with schCor(O',O_1) may be written as:
M' ≡ O'=OProduct(makeObject(C_j,S_j),OSelect(p',O_1))

The mapping M associated with schCor(O_{1F},O_1) can be then given as:

$M \equiv O_{1F}$=OProduct(makeObject(C_j,S_j),O')
= OProduct(makeObject(C_j,S_j),OProduct(makeObject(C_j,S'$_j$),OSelect(p',O_1)))
= OProduct(makeObject(C_j,$S_j \cap$S'$_j$),OSelect(p',O_1))

Rule 4: *Context Conditioning Rule*, i.e. semCondition($Cntxt_1$,semPro(O_{1F},O_1))

Rule 4.1: *Empty Context Conditioning Rule*, i.e. $Cntxt_1 = <>$

semCondition($Cntxt_1$,semPro(O_{1F},O_1)) = semPro(O_{1F},O_1)

Rule 4.2: *Constraint Conditioning Rule*, i.e. $Cntxt_1$ = glb(<(C_j,S_j)>,$Cntxt_2$)

semCondition($Cntxt_1$,semPro(O_{1F},O_1))
= semConstrain(<(C_j,S_j)>,semCondition($Cntxt_2$,semPro(O_{1F},O_1)))
Π_{Cntxt_1}(semConstrain(<(C_j,S_j)>,semCondition($Cntxt_2$,semPro(O_{1F},O_1))))
= strConstrain(map_{O_2}(C_j,A_j),S_j,Π_{Cntxt_2}(semCondition($Cntxt_2$,semPro(O_{1F},O_1))))

Rule 4.3: *Context Conditioning and semCombine Rule*, i.e.
semCondition($Cntxt_1$,semCombine(C_i,semPro(O',O_1),semPro(O",O_i)))

= semCombine(C_i,semCondition($Cntxt_1$,semPro(O',O_1)),
semCondition($Cntxt_1$,semPro(O",O_i)))
Π_{Cntxt_1}(semCombine(C_i,semCondition($Cntxt_1$,semPro(O',O_1)),
semCondition($Cntxt_1$,semPro(O",O_i))))
=
strCombine({map_{O_1}(C_i,A_i),map_{O_i}(C_i,A'$_i$)},Π_{Cntxt_1}(semCondition($Cntxt_1$,semPro(O',O_1))),
Π_{Cntxt_1}(semCondition($Cntxt_1$,semPro(O",O_i))))

Rule 5: *semCombine Rule*, i.e. Cntxt = glb(<(C_i,$O_i \circ C_{ass}$(O_i,O_1))>,$Cntxt_1$)

semPro(O_{1F},O_1) = semConstrain(<(C_i,$O_i \circ C_{ass}$(O_i,O_1))>,semPro(O',O_1))
= semCombine(C_i,semPro(O',O_1),semCondition(C_{ass}(O_i,O_1),semPro(O_{iF},O_i)))
where semPro(O',O_1) is defined *wrt* $Cntxt_1$ and O' is a temporary object obtained by applying all the constraints in $Cntxt_1$ to O_1

Π_{Cntxt}(semCombine(C_i,semPro(O',O_1),semCondition(C_{ass}(O_i,O_1),semPro(O_{iF},O_i))))
= strCombine({map_{O_1}(C_i,A_i),map_{O_i}(C_i,A'$_i$)},Π_{Cntxt_1}(semPro(O',O_2)),
$\Pi_{C_{ass}(O_i,O_1)}$(semCondition(C_{ass}(O_i,O_1),semPro(O_{iF},O_i))))
= strCombine({map_{O_i}(C_i,A'$_i$),map_{O_1}(C_i,A_i)},schCor(O',O_1),schCor(O",O_i))
where O" is a temporary object obtained by applying all the constraints in C_{ass}(O_i,O_1) to O_{iF}
and the mappings M' and M" associated with schCor(O',O_1) and schCor(O",O_i) are given as:
M' $\equiv$ O'=OSelect(p',O_1) M" $\equiv$ O"=OSelect(p",O_i)

Rule 5.1: *New Constraint and Existing Attributes*, i.e. $C_i \notin Cntxt_1$,map_{O_i}(C_i,A'$_i$) and map_{O_1}(C_i,A_i) exist

strCombine({map_{O_1}(C_i,A_i),map_{O_i}(C_i,A'$_i$)},schCor(O',O_1),schCor(O",O_i))
$M \equiv O_{1F}$=OJoin(g(A_j,A'$_j$),O',O")
=OJoin(g(A_i,A'$_i$),OSelect(p',O_1),OSelect(p",O_i))

Rule 5.2 *Coordinate Composition Rule*, i.e. C_i = compose($C_{i,1}$,$C_{i,2}$)

The composition of attributes is as follows:

map_O(C_i,X) = map_O(compose($C_{i,1}$,$C_{i,2}$),compose(X_1,X_2))
= compose(map_O($C_{i,1}$,X_1),map_O($C_{i,2}$,X_2))
Let map_{O_1}(C_i,A_i) = compose(map_{O_1}($C_{i,1}$,$A_{i,1}$),map_{O_1}($C_{i,2}$,$A_{i,2}$))
Let map_{O_i}(C_i,A_i) = compose(map_{O_i}($C_{i,1}$,A'$_{i,1}$),map_{O_i}($C_{i,2}$,A'$_{i,2}$))

The Mapping M associated with schCor(O_{1F},O_1) is given as:

$M \equiv O_{1F}$=OJoin(g(<$A_{i,1}$,$A_{i,2}$>,<A'$_{i,1}$,A'$_{i,2}$>),O',O")
=OJoin(g(<$A_{i,1}$,$A_{i,2}$>,<A'$_{i,1}$,A'$_{i,2}$>),OSelect(p',O_1),OSelect(p",O_i))

A.2 Taxonomies of schematic conflicts

In this section we enumerate the various types of schematic/representational conflicts identified by us in the taxonomy proposed in this paper. We take a representative sample of the multidatabase literature in this area and show the relationship of their work with ours by means of a table. We believe this paper provides a more complete enumeration of the various types of conflicts and their definitions.

Schematic Conflicts	[DH84]	[CRE87]	[SPD92]	[SK92]	[KCGS93]	[HM93]
Domain Incompatibilities		β	α	α		
Naming Conflicts	β	α	β	β	β	α
Data Representation Conflicts		α		β	β	
Data Scaling Conflicts	β		α	β	β	β
Data Precision Conflicts				β	β	
Default Value Conflicts				β	β	α
Attribute Integrity Constraint Conflicts			α	β	β	α
Entity Definition Incompatibilities		β		α	α	
Database Identifier Conflicts	α			β	β	
Naming Conflicts	β	α		β	β	β
Schema Isomorphism Conflicts	α	α	β	β	β	α
Missing Data Item Conflicts	β			β	β	α
Data Value Incompatibilities	α			α	α	
Known Inconsistency	β			β	β	
Temporary Inconsistency	β			β	β	
Acceptable Inconsistency				β		
Abstraction Level Incompatibilities	α			α	α	
Generalization Conflicts	β		β	β	β	β
Aggregation Conflicts	β		α	β	β	β
Schematic Discrepancies				α		
Data Value Attribute Conflict				β		
Attribute Entity Conflict	α		β	β	β	
Data Value Entity Conflict				β		

Table 3: Comparison of the Types of Conflicts

Legend:

- We use the symbol α to denote that the reference has an informal discussion of the schematic conflict.
- We use the symbol β to denote that the schematic conflict has been defined formally.