Open access • Journal Article • DOI:10.1007/S007780050029

Semantic and schematic similarities between database objects: a context-based approach — Source link

Vipul Kashyap, Amit P. Sheth

Institutions: Rutgers University, University of Georgia

Published on: 12 Dec 1996 - Very Large Data Bases

Topics: Semantic similarity, Semantic computing, Semantic grid, Description logic and Schema (psychology)

Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach

Vipul Kashyap

Amit P. Sheth
Wright State University - Main Campus, amit@sc.edu

Semantic and Schematic Similarities between Objects in Databases: 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

February 16, 1995

Abstract

In a multidatabase system, schematic conflicts between two objects are usually of interest only when the objects have some semantic similarity. In this paper we try to reconcile the schematic and semantic perspectives. We propose the concept of *semantic proximity*, which is essentially an *abstraction/mapping* between the domains of the two objects associated with the *context of comparison*. The need for making explicit the meaning and use of an object provides the motivation for explicit representation of the semantic fulcrum, i.e., the context of comparison. A partial representation of context as a collection of contextual coordinates and their values is proposed. The semantics of the specificity relationship between two contexts is defined. The contexts are organized as a meet semi-lattice[1]. Associated operations like the greatest lower bound (*glb*) of two contexts and other operations are also defined.

These operations along with the information on the type of abstractions used to relate two object classes form the basis of a semantic taxonomy. The *schematic and data conflicts* between object classes are enumerated and classified. We then try to achieve the reconciliation of the semantic and schematic perspectives by discussing *possible semantic similarities* between two object classes that have various types of schematic and data conflicts.

We introduce a uniform formalism called *schema correspondences* to represent structural similarities between the object classes. At the *semantic level* the intensional description of the object classes in a database is provided by the context expressed in a description logic like language. The schema correspondences use a modified object algebra to store mappings from the semantic level to the actual data organization in the databases and are associated with the respective contexts. We again try to achieve the reconciliation of the semantic and schematic perspectives by modeling the schema correspondences as the projection of semantic proximity with respect to (*wrt*) context. Changes in the context lead to changes in the schema correspondences. An algebra to model these is also presented and explained with the help of illustrative examples.

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

[1]A meet semi-lattice is a set in which there exists a partial order among the elements and for any two elements, there exists a greatest lower bound.

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 [She91b].

Three of the best known approaches to deal with multiple databases are tightly-coupled federation, loosely-coupled federation, and interdependent data management [SL90][She91a]. A critical task in creating a tightly-coupled federation is that of schema integration (e.g., [DH84]). A critical task in accessing data in a loosely-coupled federation [LA86, HM85] is to define a view over multiple databases or to define a query using a multidatabase language. A critical task in interdependent data management is to define multidatabase interdependencies [RSK91].

In performing any of these critical tasks, and hence in any approach to interoperability of database systems, the fundamental question is that of identifying objects in different databases that are semantically related, and then resolve the schematic differences among semantically related objects. In this paper, we are interested in reconciling the semantic and the schematic perspectives.

We characterize the degree of semantic similarity between a pair of objects using the concept of **semantic proximity** [SK92] which is introduced in Section 2.1 of this paper. The definition of this concept is based on the premise that in order to capture the semantic similarity between objects, it is essential to associate the **abstractions/mappings** between them with the **context** in which they are being compared. Other researchers in the field of multidatabases have also made similar observations in principle, albeit different in details in [ON93, SSR92, YSDK91]. This association of context with abstractions represents the first step in achieving the reconciliation between the semantic and schematic perspectives.

The two perspectives of the semantics of an object, viz., the *meaning* and *use* of an object are identified. We take cues from fields like linguistics, cognitive psychology, AI, Databases and programming languages [Woo85, BW85, SSR92, Tho89, BN75, MC91] and explore research in these areas based on these two perspectives. This helps us provide a rationale for **context** being the pivot of the semantic proximity between two objects. The computational benefits of a context representation in a multidatabase system are also discussed. Here we draw parallels between AI/Knowledge-based systems and Multidatabase systems [Sho91].

We have proposed a partial representation of context for semantic interoperability in multidatabase systems. This is important in order to have automatic ways of comparing and manipulating contexts. There have been attempts to represent similarity between two objects in databases [LNE89, SSR92], linguistics [CMG90], text retrieval [VD92] and clustering [ML92]. We have abstracted out the commonality in these approaches in our proposed representation of context as a collection of contextual coordinates and their values. Context is modeled as an intensional description of object classes and as a collection of constraints which the object classes must satisfy. The meaning of the contextual coordinates and their values are explained by expressing the context in a description logic like language [BS85, BBMR89].

Based on the proposed representation of context, we define the specificity relationship between two contexts. It is also the case that two contexts may not be comparable to each other. Thus the specificity relationship induces a partial order on the contexts. The contexts are then organized as a meet semi-lattice where there exists a greatest lower bound between any two contexts. A definition of the specificity relationship and the *glb* operation is presented. Other operations on contexts are also defined. The semantic proximity descriptor consists of context as the first component and abstraction as the second component. 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.

While there is a significant amount of literature discussing schematic differences, work on semantic issues (e.g., [Ken91]) in the database context is scarce. 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 2 in Appendix A.5). However, purely schematic considerations do not suffice to determine the similarity between objects [FKN91][SG89]. In this paper we try to reconcile the two perspectives by enumerating the possible semantic similarities between object classes and types having schematic heterogeneities.

Even though the representation of semantic better enables us to represent the similarities between the various object classes, 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 object classes. 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.3. Changes in context lead to changes in schema correspondences. These changes are captured in an algebra defined specially for this purpose and presented in the Appendix A.4.

The overall organization of the paper is as follows. In Section 2 we present a model to represent semantic similarities among object classes. In Section 3 we discuss the rationale for explicit identification and representation of context in a multidatabase environment. In Section 4 we discuss an explicit, though partial, representation of context and operations for reasoning about and manipulating the context representations. In Section 5 a taxonomy of the various types of possible semantic similarities between the various object classes is presented. In Section 6 we discuss a broad class of schematic differences and the possible semantic similarities between object classes having those differences. In Section 7 we define a uniform formalism for representation of structural similarity. It is associated with the context and is defined as the projection of the semantic similarity. In Section 8 we illustrate with the help of examples how changes in context change the semantic and structural similarity between two object classes. Conclusions and future work are presented in Section 9.

2 Semantic Similarities between Objects

In this section, we discuss the concept of *semantic proximity* to characterize **semantic** similarities between objects. A classification of semantic similarities between objects is presented based on this concept is presented in Section 5. We distinguish between the *real world*, and the *model world* which is a representation of the real world. It is our endeavor to capture some of the important semantic information about the real world and represent it in the model world.

In characterizing the similarity between objects based on the semantics associated with it we have to consider the *meaning* and the *use* of the objects (sometimes not only limited to current, but also intended and/or future). Wood [Woo85] defines semantics to be **the scientific study of the relations between signs and symbols and what they denote or mean**. A complementary perspective would be the **scientific study of how signs and symbols are used**. It is not possible to completely define what an object denotes or means in the model world [SG89]. We consider these to be aspects of *real world semantics* (RWS) of an object[2]. What constitutes meaning is hard to define but we can take cues from the study of semantics undertaken in philosophy,

[2]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.

linguistics, cognitive psychology, artificial intelligence, information systems and data modeling. What constitutes use is identified by users, administrators and application programmers and is often modeled as meta-information in data models. An important issue here is the environment (viz. application, query, information system) in which the object is used. In Section 3 we take the above mentioned perspectives on semantics to explain the rationale for representation of context in the model world.

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 RWS of an object cannot be captured using mathematical formalisms. We need to understand and represent more knowledge in order to capture the semantics of the relationships between the objects. The knowledge should be able to capture and the representation should be able to express 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 [Guh90].

This viewpoint has been reflected by the techniques and representational constructs used by the various practitioners in the field of multidatabases and has influenced the model for semantic proximity defined in Section 2.1. In Sections 2.2 and 2.3, we explore the above viewpoint in detail and illustrate how the various researchers have tackled these issues in their attempts to represent semantic similarity. Notable among them are the **semantic proximity** proposal by Sheth and Kashyap [SK92], the **context building** approach taken by Ouksel and Naiman [ON93], the **context interchange** approach taken by Sciore et al [SSR92] and the **common concepts** approach taken by Yu et al [YSDK91]. A detailed discussion of these can be found in [KS95].

The term object in this paper refers to an object in a model world (i.e., a representation or intensional definition in the model world, e.g., an object class definition in object-oriented models) as opposed to an entity or a concept in the real world. These objects may model information at any level of representation, viz. *attribute level* or *entity level*[3].

Our emphasis is on identifying semantic similarity even when the objects have significant representational differences [She91b]. Semantic proximity is an attempt to characterize the degree of semantic similarity between two objects using the RWS. It provides a qualitative measure (Section 5) to distinguish between the terms introduced in [She91b], *viz. semantic equivalence*, *semantic relationship*, *semantic relevance* and*semantic resemblance*. Two objects can be *semantically similar* in one of the above four ways. Semantic equivalence is *semantically closer* than semantic relationship, and so on. This is illustrated in Figure 4.

2.1 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

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.

[3]Objects at the entity level can be denoted by single-place predicates P(x) and attributes can be denoted by two-place predicates Q(x,y) [SG89].

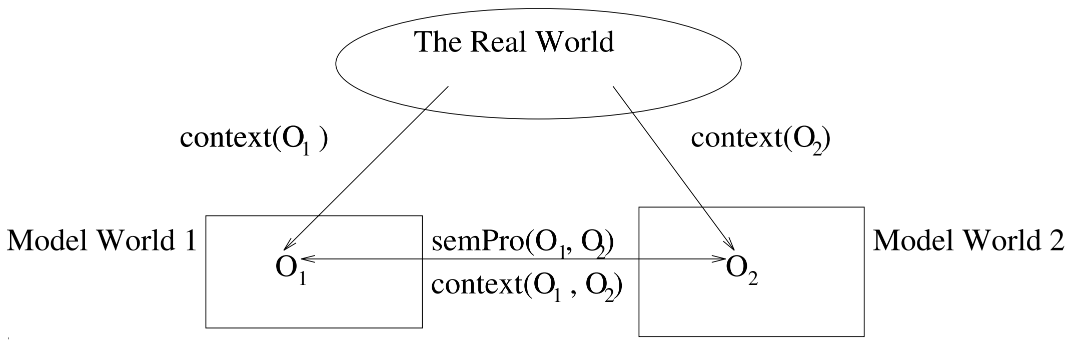


Figure 1: Semantic Proximity between two Objects

- The second component identifies the abstraction 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 in which the objects O_1 and O_2 are mapped from the real world to the model world[4]. context(O_1,O_2) refers to the context in which the objects are being compared.

2.2 Context: The semantic component

Each object has its own context. The term context in semPro refers to the context in which a particular semantic similarity holds. This context may be related to or different from the contexts in which the objects were defined. It is possible for two objects to be semantically closer in one context than in another context. The context, provides the *semantic fulcrum* of capturing and representing the object similarities. The rationale behind this viewpoint is discussed in Section 3. Some of the alternatives for representing a context in a *multidatabase system* are as follows.

- In [SK92] the concept of **semantic proximity** is proposed to characterize semantic similarity in which the context is the primary vehicle to capture the RWS.
- In [ON93], context is defined as the knowledge that is needed to reason about another system, for the purpose of answering a query and is specified as a set of assertions.
- In [SSR92], context is defined as the meaning, content, organization and properties of data and is modeled using metadata associated with the data.
- In [YSDK91] **common concepts** are proposed to characterize similarities between attributes in multiple databases. Cherchia and McConnell-Ginet [CMG90] propose that a concept may be considered to be the image of a function mapping contexts to propositions. Thus a context may be implicitly represented in the functional definitions of the concepts.

[4]We shall refer to these contexts as definition contexts of the respective objects later on in this paper.

- When using a well defined ontology, such as Cyc [Guh90], a well defined partition (called Microtheory) of the ontology can be assigned a context.
- A context may also be associated with a **database** or a group of databases (e.g., the object is defined in the context of DB1).
- The **relationship** in which an entity participates may determine the context of the entity.
- 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, a context can be thought of as a **named collection** of the domains of the Objects.
- Sometimes a context can be "hard-coded" into the definition of an object. For example, when we have the two entities EMPLOYEE and TELECOMM-EMPLOYEE, the TELECOMMUNICATIONS context is "hard-coded" in the second entity. We are interested in representing and reasoning about context as an explicit concept.

2.3 Abstractions/Mappings: The Structural Component

Abstraction here refers to the relation between the domains of the two objects. Mapping between the domains of objects is the mathematical tool used to express the abstractions. However, since abstractions by themselves cannot capture the 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 (refer to Section 2.1).
- 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.

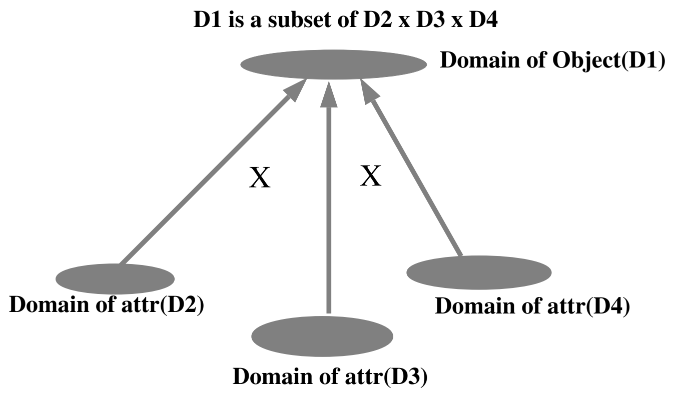


Figure 2: Domain of an Object and it's Attributes

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 two objects.

NONE This is used to denote that there is no mapping defined between two semantically related objects.

In Section 7.1, we define the concept of *schema correspondences* to express the abstractions in a uniform formalism.

2.4 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, viz. union and intersection of domains.

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.

2.5 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 Rationale for representing Context in the model world

The context of an object is the primary vehicle to capture the RWS of the object. The context in which two objects are being compared and the abstraction/mapping associated helps to capture the semantic aspect of the relationship between two objects (Figure 1). In Section 2.2, we have identified the context as the *semantic fulcrum* of the semantic proximity proposed in Section 2.1. In this section, we explicate the relationship between the real world semantics of an object in the model world and the context in which the object is defined (or compared with another object). We argue for the need of representing context by showing how purely structural representations are inadequate. We also look at the semantics of an object from the perspectives of the meaning and use of objects to justify the need for the representation of context.

3.1 Inadequacy of purely Structural Representations

It has been suggested in Sheth and Gala/Kashyap [SG89][SGN93] and Fankhauser et al. [FKN91] that the ability to capture and represent the structure of an object does not help capture the real world semantics of the object. Fankhauser et al. [FKN91] suggest that it is not possible to provide a structural and hence a mathematical definition of the complex notion of real world semantics. Hence when one tries to capture the semantic similarities between two objects, the existence of mappings between the domains of the objects and the similarity in their structures is not enough to guarantee semantic similarity. This is illustrated with the help of an example in [SG89].

Example:
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 descriptors such as:

```
Uniqueness
Lower and Upper Bound
Domain and Scale
Static and Dynamic Integrity Constraints
Security Constraints
Allowable Operations
```

The attribute theory proposed by them says that if there exist mapping functions that satisfy certain properties between the domains of the two attributes, then the two attributes can be said to be equivalent. However, while the descriptors (which help generate the mappings), are sufficient to establish the structural equivalence of the attributes and are a necessary condition for the equivalence of the attributes, they are not sufficient to establish 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. There should then be some way to denote their lack of semantic equivalence. We propose that the definition of mappings between the domains of the two objects be made *wrt* to a context. As defined later, if two objects are semantically equivalent, then **it should be possible to define mappings wrt all known and coherent contexts and the definition contexts**[5] **should not be incoherent wrt each other**. Semantic Equivalence is discussed in detail in Section 5. Since it is not possible to define mappings between *person-name* and *department-name* *wrt* all known

[5]The definition context of an object and the coherence of contexts are formally defined in Section 4

contexts, we cannot call them equivalent. In this way, **context** can help capture an aspect of real world semantics better than purely structural mapping methods.

Example:
An Abstraction by itself does not capture the semantics of a relationship. Consider the two object classes defined in different databases.
STUDENT(Id#, Name, Grade)
DEPARTMENT(Num, Name, Address)
Let Domain(Id#) = Domain(Num) = {123, 456, 789}

Thus, it is possible to construct a mapping between Id# and Num, two semantically unrelated objects. This mapping however shall not be possible if we represent the contexts of Id# and Num as the contexts associated with STUDENT and DEPARTMENT respectively.

3.2 Semantics: The meaning of an object

We take cues from the fields of linguistics, programming languages and knowledge representation in AI on what constitutes meaning, and how the meaning of an object can be best expressed in an explicit representation.

In linguistics [Woo85], the primary interest in semantics has been in characterizing the fact that the same sentence can sometimes mean different things. A knowledge engineer in the field of knowledge representation in AI [BW85], on the other hand, is usually interested in a (semantic) description that must be able to represent partial knowledge about an entity and accommodate multiple descriptors which can describe the associated entity from different viewpoints.

In the area of programming languages, one approach to semantics has been to define a mathematical object for each language entity and a function that maps instances of that entity onto instances of that mathematical object. Because the objects are rigorously defined, they represent the exact meaning of their corresponding entities. This approach is called the *denotational semantics approach* because the objects denote the meaning of the syntactic entities.

Our goal in representation of semantics is to ensure *semantic interoperability* between the various databases. This means that the databases must share a common meaning for the objects they share [SSR92]. In a multidatabase environment, each database makes its own assumptions about the meaning of an object. Thus it is necessary to understand and share the information about the assumptions made while defining an object in each database. The context associated with the database is a good tool to capture information about the design assumptions of each database. In a multidatabase environment, the contents of a database can be meaningful given a context and the meaning/significance can be looked at in terms of an interpretation in the context [Tho89].

We observe a commonality in diverse fields of research when it comes to representing the meaning of an object. The commonality being that the same sentence/entity/object can have different meanings/descriptions/design assumptions. We propose that in either case, it is the **context** of the sentence/entity/object which determines the applicable meaning/descriptor/assumption. When two objects are compared for semantic similarity, it is the context of comparison which determines appropriate meanings/descriptors/assumptions of each of the sentences/entities/objects.

3.3 Semantics: The use of an object

This viewpoint of the semantics of an object, i.e., the way it is used, is typically held in the field of programming languages and artificial intelligence. In programming languages, the semantic specifications are often defined in terms of the procedures or operations to be carried out on the objects. The semantics of a language is described by specifying the behavior of an abstract processor that executes programs written in that language and is referred to as *operational semantics*. The view suggested in AI is that one memory schema refers to another only through the use of a description which is dependent on the context of the original reference [BN75].

In the area of linguistics and cognitive psychology, experiments have borne a strong relationship between semantic and contextual similarity [MC91]. This has led to the belief that semantic similarity is a function of the contexts in which an object is used and that the contextual representation of an object is the knowledge of how that object is used. Hence it follows that the contextual representation is an abstract cognitive structure that accumulates the attributes common to all the contexts in which an object is used [MC91].

Contexts in which the objects are accessed or in which operations are executed on the objects are critical in determining how that object is interpreted and used. Every database system and application program makes assumptions about the use of the objects and their stored extensions. In a multidatabase setting, the objects have to interoperate with other similar and not so similar objects. Since our goal is to achieve *semantic interoperability* in multidatabase systems, these assumptions should be made explicit as far as possible, and we should have a methodology to deal with such an explicit representation.

Keeping the above points in mind, context provides us with a tool for the *migration of object semantics* out of the respective applications and individual databases. Information needed to determine semantic similarity or resolve semantic conflicts between objects is hard coded implicitly in the code written to do the same. Also, mappings defined between various objects in different databases might be dependent on the usage. Thus, explicit representation of context can help to solve the problems associated with interoperability by decoupling various applications and databases from each other.

3.4 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 those and multidatabase systems and context representation might enable us to handle information in a multidatabase system more cleanly and efficiently. Some benefits are as follows:

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 7.2.

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 [KS94] 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 more detail in Section 6.3.

Flexible semantics: A big fallout of associating abstractions/mappings with the context in the semantic proximity model (Section 2.1) 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.

4 Explicit context representation in a multidatabase environment

For semantic interoperability in multidatabase systems, it is important to have appropriate representations of context (which should capture the semantics of the information in the databases) and develop a practical framework for:

- Semi-automatic ways of constructing contexts from an ontology or a collection of metadata.
- 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 partial context specification can be used by humans to decide whether the context for modeling of two objects is the same or different or non-comparable. In this section we propose such a partial representation of context. It must be noted that in order to judge the semantic similarity between any two objects, a partial representation should suffice.

4.1 A partial Context representation

There have been attempts to represent the similarity between two objects in databases. In [LNE89] (see Example in Section 3.1), a fixed set of descriptors define essential characteristics of the attribute and are used to generate mappings between them. However, we have already demonstrated that they do not guarantee semantic similarity. Thus, any representation of context which can be described by a fixed set of descriptors is not acceptable.

In [SSR92], context is represented as a collection of meta-attributes and is associated with each data value. The meta-attributes are also known as context coordinates in linguistics [CMG90], thematic roles in text retrieval [VD92] and code words by researchers in clustering techniques [ML92]. One commonality in these approaches is that the set of meta-attributes is not fixed but dynamically chosen to model the characteristics of the application domain in question. We recognize the fact that it is not possible a priori to determine all possible meta-attributes which would completely characterize the semantics of the application domain. Hence we admit to a *partial* representation of context.
Based on the above discussion, we represent a 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 with examples the meaning of the symbols C_i and V_i and how they can be used. We shall also explain each example by using a description logic like language. This language has been used in and is exemplified by [BS85, BBMR89, Mac87, PS84, vLNPS87, KBR86]. Using this language, 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. This can 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 class 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 an ontology.
- C_i may model an implicit assumption in the design of a database. The value of the contextual coordinate V_i may express some constraints which capture the assumptions in the design of the database.
- C_i may or may not be associated with an attribute A_j of an object class O in the database. This is discussed in detail in Section 7.2 of this paper.

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 class, a 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 in Section 4.1.3) 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 (**FILLS** author) (**FILLS** designee))

- V_i can be a set.
  - The set may be an enumeration of symbols from the ontology.
  - The set may be defined as the extension of an object class or as elements from the domain of a type defined in the database.
  - The set may be defined by defining constraints on pre-existing sets.

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

  - **Deptypes** is a type defined in the database.
  - The symbols restypes, employer, article are taken from the ontology.
  - The definition context (defined in Section 4.1.1) expresses a semantic association between EMPLOYEE and PUBLICATION which may not be captured in the database. If the contextual coordinates employer and article are not associated with any attribute of the object class EMPLOYEE, the context models information not modeled in the database. The extra information which can be thus modeled is bounded by the ontology. This is discussed in detail in Section 7.2.

- 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.
  - The constraints would apply to the set, type or object class the variable X would unify with.
  - This is called the constraint context and is defined in Section 4.1.4.

  Example:
  Suppose we want all the articles which contain the substring "abortion" in them. This can be expressed in the following query context:
  C_q = <(article, X∘ <(title, {y|substring(y) = "abortion"})>)>
  C_q = <(article, X∘Cntxt)>
  where ∘ denotes association of a context with a variable and
  Cntxt = <(title, {y|substring(y) = "abortion"})>
  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 class associated with a context.
  - This is called the association context and is defined in Section 4.1.2.
  - This may be used to express semantic dependencies between object classes which may not be modeled in the database. This is discussed in detail in Section 7.2.
  - The context also provides us with a mechanism to correlate information. This is discussed in detail in Section 8.

  Example:
  Suppose we want to represent information relating publications to employees in a database. Let PUBLICATION and EMPLOYEE be object classes 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 class and
  Cntxt = <(affiliation, {research})>

The same thing can be expressed in a DL as follows:

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

It may be noted that in the case the object class HAS-PUBLICATION is not defined in the database, the context models information not modeled in the database. This is discussed in detail in Section 7.2.

4.1.1 Definition Context of an Object Class

Given an object class O in a database and a collection of contextual coordinates C_is from the ontology, the definition context can be represented as:

$C_{def}(O) = <(C_1, V_1) ... (C_k, V_k)>$

This can be used in the following ways:

- To specify the assumptions in the design of the object class O.
- To share only a pre-determined extension of the object class O with the federation of databases. This object class is denoted as O_F.
- The associations between the object classes stored in the database and the object classes exported to the federation are expressed using the concepts of **semantic proximity** and **schema correspondences** (defined in Section 7.1). The associations are discussed in detail in Section 7.2.

4.1.2 Association Context of Object Classes

Given object classes O and O_1 in a database and a collection of contextual coordinates C_is from the ontology, the definition context of O_1 which depends on the context of association between O and O_1, $\mathbf{C_{ass}(O_1, O)}$ can be represented as:

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

The association context can be used in the following ways:

- To represent relationships between two object classes with reference to an aspect of an application domain. This is done by associating it with the appropriate contextual coordinate.
- Different relationships between two object classes 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 class O and different (more than one) objects as a part of the definition context of the same object. Thus, the context of an object consists of it's relationships with other objects.

- The relationships being modeled may not be restricted to constructs in the data-model, though a mapping may be provided between the constructs used in the definition context and those available in the data-model.

4.1.3 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 [KS94, KS].
- Object classes and types defined in databases are also available to the user after being appropriately incorporated 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.

4.1.4 Constraint Context

This context is typically a part of the query context and is used to pose constraints on the answer returned for the query. C_{constr}(X,ANSWER) can be represented as:

C_Q = <(C_1, X$\circ C_{constr}$(X, ANSWER)) ... >

- 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 class or type definition at run time.
- The context may represent constraints on the object class or it's attributes or the contextual coordinates associated with an object class.
- 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.

4.2 Reasoning about and manipulation of contexts

We have proposed a partial representation of context in the previous section. To use this representation meaningfully in focusing on relevant information and correlating information we need the following 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 object classes defined in a particular context have to transcend [McC93] to a more specific or general context. This is discussed in detail with examples in Section 8.
- 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. The set of contexts thus forms a meet semi-lattice.

Based on the representation defined in the previous section, we give the semantics of the above operations in the following sections.

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

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

- $C, C_1, C_2, C'_1, C'_2, ...$ 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:

- $m \leq k$
- $\forall i,\ 1 \leq i \leq m,\ \exists j\ C_j \leq C'_i$[6] $\wedge\ V_j \leq V'_i$

[6]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.

4.2.2 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 on in the paper,

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

generalize(Cntxt$_1$, C$_i$) $= <(C_i, V_i)>$

compare(Cntxt$_1$, Cntxt$_2$) This operator is used to compare contexts. It essentially computes the glb of two contexts, but considers only those contextual coordinates which are common to both the contexts.

$=$ generalize(glb(Cntxt$_1$, Cntxt$_2$), overlap(Cntxt$_1$, Cntxt$_2$))

coherent(Cntxt$_1$, Cntxt$_2$) This operator determines whether the constraints determined by the values of the contextual coordinates are inconsistent or not.

4.2.3 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 the **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.

- $\text{glb}(A_1 \circ \text{Cntxt}_i, A_2) = \text{glb}(A_1, A_2) \circ \text{Cntxt}_i$
- $\text{glb}(A_i \circ \text{Cntxt}_i, A_j \circ \text{Cntxt}_j) = \text{glb}(A_i, A_j) \circ \text{glb}(\text{Cntxt}_i, \text{Cntxt}_j)$

The greatest lower bound of the contexts can now be defined as:

- $\text{glb}(\text{Cntxt}, <>) = \text{Cntxt}$
- $\text{glb}(<(C_i, V_i) \ldots >, <(C'_i, V'_i) \ldots >) = <(C_i, V_i) \ldots (C'_i, V'_i) \ldots >$
  where $C_i \neq C'_i$
- glb(Cntxt$_1$, Cntxt$_2$)
  $= \text{glb}(<(C_i, V_i) \ldots >, \text{glb}(<(C_j, V_j) \ldots >, <(C_k, \text{glb}(V_k, V'_k)) \ldots >))$
  where $C_i, C_j \notin$ overlap(Cntxt$_1$, Cntxt$_2$)
  and $C_k \in$ overlap(Cntxt$_1$, Cntxt$_2$)

An alternative equivalent representation of a context can be expressed using the glb operation described above. However, it 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) ... (C_k, V_k)> can also be represented as:

Cntxt = glb(<(C_1, V_1)>, glb(<(C_2, V_2)>, ... , glb(<(C_k, V_k)>, <>) ...))

Example:
Consider the following two contexts:
$Cntxt_1$ = <(author, EMPLOYEE∘ <(affiliation, {research})>)
(article, PUBLICATION)>
$Cntxt_2$ = <(article, X∘ <(title,{x| substring(x)="abortion"})>)>

It should be noted that:

- PUBLICATION can be assumed to have an empty context, i.e. PUBLICATION∘ <>
- article $\in$ overlap($Cntxt_1$,$Cntxt_2$)
- author $\notin$ overlap($Cntxt_1$,$Cntxt_2$)

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

In our approach to represent context, 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 believe that in a database with a large number of data values it's not feasible to do so.
- 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 object classes which cannot be captured at the extensional level. We also view the context of an object as a collection of constraints on an object class 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 [KS94, KS].
- We have also defined operations to compare the specificity of contexts, and to manipulate and reason about them. This makes it easier to perform inferences on context to support query processing against the multiple databases.
- Based on the partial order induced by the specificity relationship, we organize the contexts as a meet semi-lattice.

4.3 Issues of language and ontology in context representation

We have presented an explicit representation of context in Section 4.1 and the semantics of operations to compare and manipulate these representations in Section 4.2. In this section we discuss the issues of a language in which the explicit representation can be best expressed. We also discuss issues of ontology, i.e. the vocabulary used by the language to represent the contexts.

4.3.1 Language for context representation

In Section 4.1 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 have the ability to describe what kinds of sentences it can describe, i.e. it should be self-describing. This enables us to represent nesting of contexts to any arbitrary level as the definition context of an object class might contain association contexts with one or more object classes.
- The language should be able to express the context as a collection of contextual coordinates, each describing a specific aspect of information in the database.
- The language should have primitives (viz., 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 for performing inference on the ontology to identify the abstractions related to the ontological objects in the query context or the definition contexts of the object classes in the databases. We view ontology as the symbolic layer closest to the concepts in the real world.

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.

4.3.2 The Ontology Problem

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]. In constructing the contexts as illustrated in Section 4.1 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 4.1, 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 the each database has available to it an ontology corresponding to a specific domain. The definition and association contexts of the object classes 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 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 (viz. frames, slots) in Cyc's existing ontology [CHS91].
  - Another approach has been to exploit the semantics of a single problem domain (viz. 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.
- **Reuse 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 viz. ISBN classification for publications, botanical classification for plants etc. 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.

5 A Semantic Taxonomy

In this section we use the concept of semantic proximity defined in Section 2.1 and the context representation discussed above to define a semantic taxonomy consisting of the various types of semantic similarities between object classes. The taxonomy thus designed, is illustrated in Figure 4.

5.1 The role of context in semantic classification

The context, being identified as the pivot on which the semantic proximity depends, plays a key role in this taxonomy. Here we enumerate the possible values a context can take.

- ALL, i.e., the semPro between the object classes is being defined *wrt* all known and *coherent* comparison contexts. There should be coherence between the definition contexts of the object classes being compared and between them and the context of comparison.

- SOME, i.e., the semPro between the object classes 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 object classes. Typically we are interested in the *glb* of the context of comparison and the definition context of the object class.
  - LUB, i.e. the least upper bound[7] of the contexts of the two objects is taken. Typically, we are interested in the *lub* of the definition contexts of the two object classes when there doesn't exist an abstraction/mapping between their domains in the context of comparison.
- SUB-CONTEXTS, we might be interested in the **semPro** between two object classes 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 object classes may be defined. This is the case when the definition contexts of the objects being compared are *not coherent* with each other.

5.2 Semantic Equivalence

This is the strongest measure of semantic proximity two object classes can have. Two object classes 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 object classes 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), _>[8]

The notion of equivalence described above depends on the definition of the domains of the object classes and can be more specifically called *domain semantic equivalence.* We can also define a stronger notion of semantic equivalence between two object classes 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).

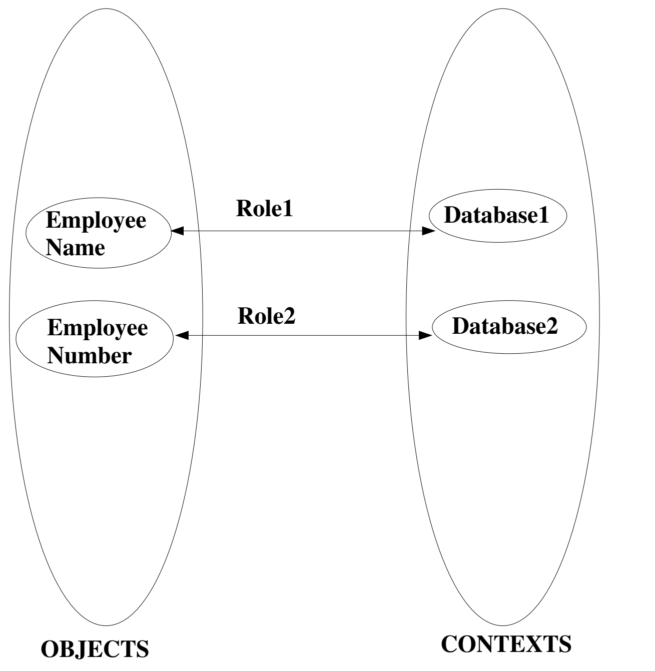
Unless explicitly mentioned, we shall use semantic equivalence to mean domain semantic equivalence.

5.3 Semantic Relationship

This is a weaker type of semantic similarity than semantic equivalence. Two object classes 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 object classes. Here we relax the requirement of a 1-1 mapping in a way that given an instance O_1 we can identify an

[7]We have not defined it for the general case. Here, we are only interested in the case where the least upper bound would consist of the union of the values of the contextual coordinates in the overlap of the two contexts.

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



Role1 = role-of(EmployeeName, Database1) = Identifier
Role2 = role-of(EmployeeNumber, Database2) = Identifier
EmployeeName in Database1.Identifier
EmployeeNumber in Database2.Identifier
Thus, Role1 = Role2

Figure 3: Roles played by objects in their contexts

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 = partial many-one value mapping, generalization, or aggregation

5.4 Semantic Relevance

We consider two object classes 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 object classes may be semantically relevant in one context, but not so in another. Object classes can be related to each other using any abstraction.

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

5.5 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 object classes cannot be related to each other by any abstraction in any context. Hence, the exact nature of semantic proximity between two object classes is very difficult to specify. In this case, the user may be presented with extensions of both the object classes. 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.

5.5.1 Role played by an Object Class in a Context

This refers to the relationship between an object class and the semantic context to which it belongs. We characterize this relationship as a binary function, which has the object and it's 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 4.1 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) = <(Classes, {Employee, ... }) (Identifiers, {Name, ...})>
C_{def}(Database2) = <(Classes, {Employee, ... }) (Identifiers, {Number, ...})>
lub(C_{def}(Database1), C_{def}(Database2))
= <(Classes, {Employee_1, Employee_2, ...}) (Identifiers, {Name, Number, ... })>

Thus role-of(Name, C_{def}(Database1)) = role-of(Number, C_{def}(Database2)) = Identifier
Since Name, Number $\in$ Identifiers $\wedge$ Identifiers $\in$ lub(C_{def}(Database1), C_{def}(Database2))

5.5.2 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 $O_1 \circ \text{Cntxt}_1$ and $O_2 \circ \text{Cntxt}_2$
and coherent($C_{def}(O_1)$,$C_{def}(O_2)$)
and SOME(LUB) denotes a context defined as follows:
where context = lub(Cntxt_1, Cntxt_2)
and $D_1 \neq D_2$
and role-of(O_1, context) = role-of(O_2, context)

5.6 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, NEG, (D_1, D_2), _>
where $C_{def}(O_1)$ and $C_{def}(O_2)$ are incoherent with each other

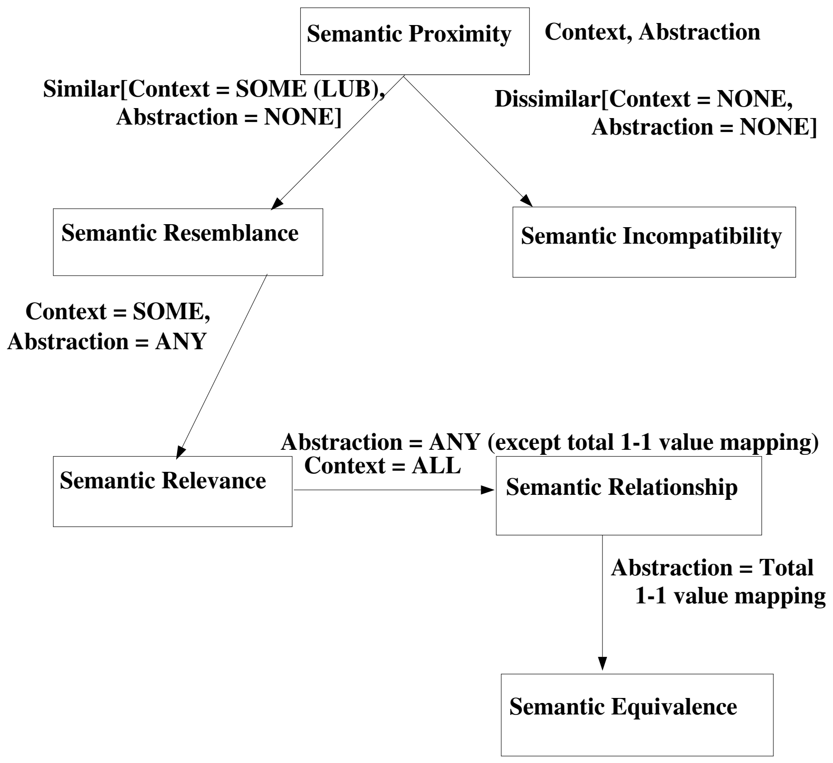

Semantic Proximity
Context, Abstraction
Similar[Context = SOME (LUB), Abstraction = NONE]
Dissimilar[Context = NONE, Abstraction = NONE]
Semantic Resemblance
Semantic Incompatibility
Context = SOME, Abstraction = ANY
Semantic Relevance
Abstraction = ANY (except total 1-1 value mapping) Context = ALL
Semantic Relationship
Abstraction = Total 1-1 value mapping
Semantic Equivalence

Figure 4: Semantic Classification of Object Similarities

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)

6 Schematic Heterogeneities in Multidatabases

In this section we deal with a broad class of schematic differences and the possible semantic similarities with between the object classes having those 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 5). 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 tie between the semantic aspects defined above and the structural aspects.

6.1 Domain Incompatibility

In this section we discuss the incompatibilities that arise between two domain types when they are differing definitions of semantically similar attribute domains. We refine the broad definition of this incompatibility given in [CRE87].

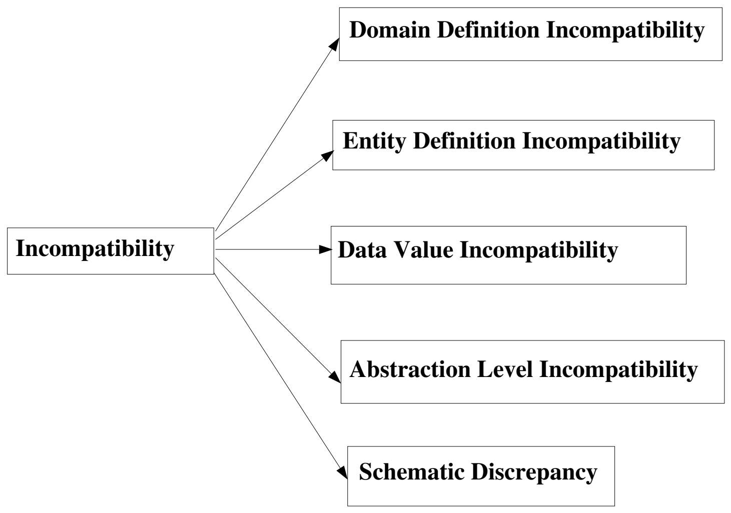

Incompatibility
Domain Definition Incompatibility
Entity Definition Incompatibility
Data Value Incompatibility
Abstraction Level Incompatibility
Schematic Discrepancy

Figure 5: Schematic Heterogeneities

6.1.1 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)
STUDENT.Id# and TEACHER.SS# 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)
STUDENT.Id# and BOOK.Id# are homonyms.
```

One alternative of defining the definition contexts of the two domain types (which are defined in two different databases) is as follows:

C_{def}(STUDENT.Id#) = <(identifies, AnimateObject[9])>
C_{def}(BOOK.Id#) = <(identifies, InAnimateObject[10])>

[9]Obtained from the ontology of the database.
[10]Obtained from the ontology of the database.

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

Table 1: Mapping between Marks and Grades

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

6.1.2 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.*

6.1.3 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.*

6.1.4 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 coarse 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}.

There may be a many-one mapping from Marks to Grades. 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 identifying the precise numerical score, is typically not possible. In such cases, the domain types may be considered *semantically related.*

6.1.5 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[11])>
When the semPro is *projected* wrt the Context, it maps to different ages in the different databases. The projection operation is discussed in detail in Section 7.2.

6.1.6 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 : $\text{Age}_1 \leq 18$
C2 : $\text{Age}_2 > 21$
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 this case these types are associated with definition contexts of their respective databases in which they exist, i.e. $\text{Age}_1 \circ C_{def}(\text{Database}_1)$ and $\text{Age}_2 \circ C_{def}(\text{Database}_2)$
$C_{def}(\text{Database}_1)$ = <(timePeriod, {Age, Duration, ...})>
$C_{def}(\text{Database}_2)$ = <(timePeriod, {Age, RacePerformance, ...})>

semPro(Age$_1$, Age$_2$) = <SOME(LUB), NONE, (D$_1$, D$_2$), _>
where SOME(LUB) denotes a context defined as follows:
where context = lub($C_{def}(\text{Database}_1)$, $C_{def}(\text{Database}_2)$)
and $D_1 \neq D_2$
and role-of(Age$_1$, context) = role-of(Age$_2$, context) = timePeriod.

Hence, they may be considered to have a *semantic resemblance* to each other.

[11] Obtained from the ontology of the database.

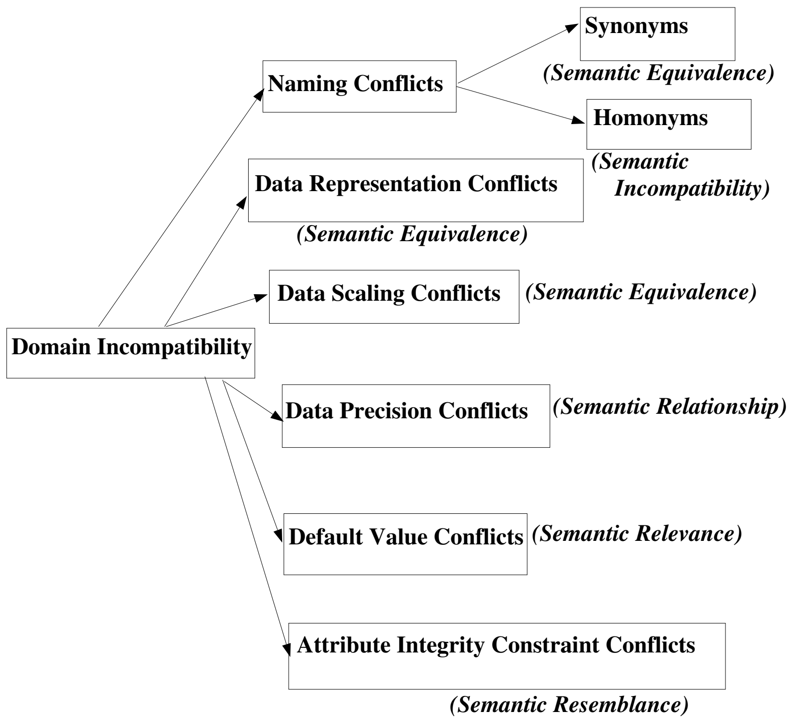


Figure 6: Domain Incompatibility and the likely types of semantic proximities

6.2 Entity Definition Incompatibility

In this section we discuss the incompatibilities that arise between two object classes 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].

6.2.1 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 SS#$\circ C_{def}$(Database1) and Name$\circ C_{def}$(Database2)

C_{def}(Database1) = <(Classes, {STUDENT1, ... }) (Identifiers, {SS#, ...})>
C_{def}(Database2) = <(Classes, {STUDENT2, ... }) (Identifiers, {Name, ...})>
and SOME(LUB) denotes a context defined as follows:
and context = lub(C_{def}(Database1), C_{def}(Database2))
and role-of(SS#, context) = role-of(Name, context) = Identifiers

6.2.2 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 object classes 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 6.1.1, their definition contexts can be modeled in a way that they are incoherent wrt each other. Thus these object classes may be be considered *semantically incompatible*.

6.2.3 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 6.1.4 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 6.1.4 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 object classes may be considered *semantically related*.

semPro(Instructor_1, Instructor_2)
= <ALL, {M_{ID}, M_1}, ({$D_{1,ID}$, $D_{1,2}$, $D_{1,3}$}, {$D_{2,ID}$, $D_{2,2}$}), _>
where M_{ID} is a total 1-1 value mapping between $D_{1,ID}$ and $D_{2,ID}$ 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 $D_{1,2} \cup D_{1,3}$ and $D_{2,2}$.

6.2.4 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 Section 6.2.3. 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 6.1.5 which is an *attribute level conflict* whereas the conflict discussed here is an *entity level conflict.* The object classes may be considered *semantically relevant* as proposed below.

The definition contexts of the two object classes 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:

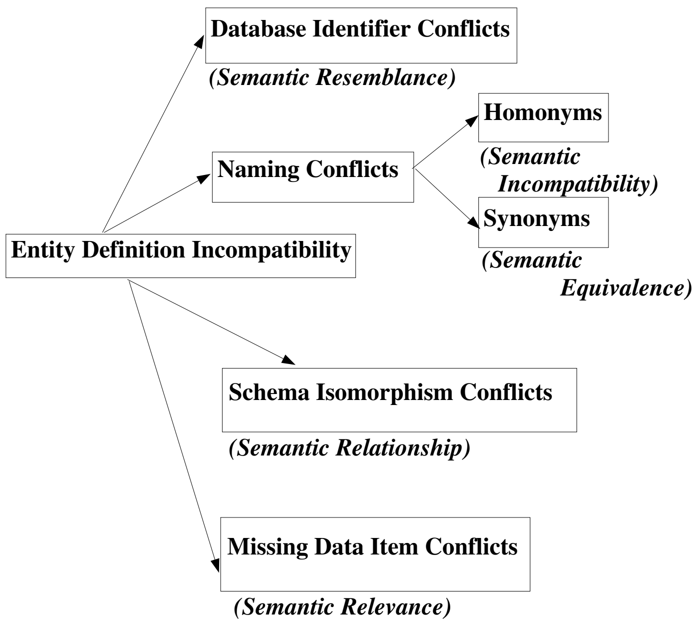
glb(C_{def}(STUDENT), C_{def}(GRAD-STUDENT)) = <(types, {graduate})>
The abstraction is then computed by conditioning the original student abstraction wrt this new context. This operation is formally defined in Section 8.

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

6.3 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 6.1.5) and attribute integrity constraint conflicts (Section 6.1.6) 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.

Example:



Database Identifier Conflicts
(Semantic Resemblance)
Homonyms
(Semantic Incompatibility)
Naming Conflicts
Synonyms
(Semantic Equivalence)
Entity Definition Incompatibility
Schema Isomorphism Conflicts
(Semantic Relationship)
Missing Data Item Conflicts
(Semantic Relevance)

Figure 7: Entity Definition Incompatibilities and the likely types of semantic proximities

```
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.
```

6.3.1 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 and hence the objects may be 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)).

6.3.2 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*[12]. 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 projected wrt context gives the mapping defined below. This is discussed in detail in Section 7.2.

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 $S_2(t + \Delta t) = S_1(t)$.

6.3.3 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 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])>

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 $\text{perturb}(S_1, \epsilon) = S_2$
and ϵ is the discrepancy in the state of the two objects.

6.4 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

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

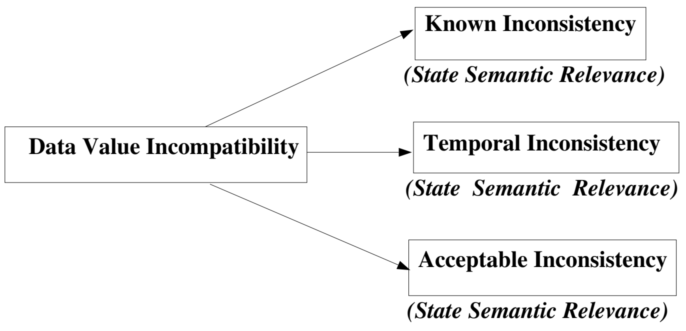

Known Inconsistency
(State Semantic Relevance)
Data Value Incompatibility
Temporal Inconsistency
(State Semantic Relevance)
Acceptable Inconsistency
(State Semantic Relevance)

Figure 8: Data value incompatibilities and the likely types of semantic proximities

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.

6.4.1 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 object classes 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)) = <(types, {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.* This operation is formally defined in Section 8.

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

6.4.2 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.

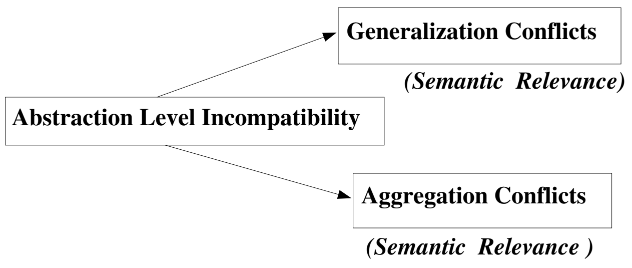


Figure 9: Abstraction level incompatibilities and the likely types of semantic proximities

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, for every ship that is a member of the convoy.
```

In this case there is a mapping in one direction only, i.e., the 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, ...)>
where weight = average(shipweight)
C_{def}(SHIP) = <(shipweight, ...) (shiplocation, ...)>
where shiplocation = location
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.*

6.5 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 6.3 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].

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) . . . },
  $\vdots$

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).

6.5.1 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 set 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. 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.

6.5.2 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 6.2. 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 (object 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))

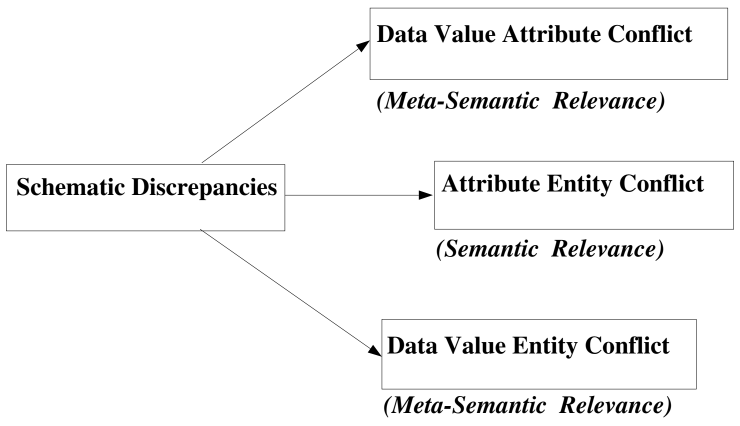

Data Value Attribute Conflict
(Meta-Semantic Relevance)
Schematic Discrepancies
Attribute Entity Conflict
(Semantic Relevance)
Data Value Entity Conflict
(Meta-Semantic Relevance)

Figure 10: Schematic Discrepancies and the likely types of semantic proximities

and D_1 = Domain(O_1)
and D_2 = Domain(Identifier(O_2)).

6.5.3 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.

7 Structural Similarity: A component of Semantic Similarity

In this section we propose a uniform formalism for representation of structural similarities between objects which are called **schema correspondences**. These are associations between object classes and types defined in the various databases and can be expressed using a modified object algebra discussed in Section 7.1. The schema correspondences so defined are however a part of the semantic proximity between the two object classes or types and are dependent on the context in which the semantic proximity is defined. This is discussed in detail in Section 7.2.

7.1 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

$$\mathbf{schCor(O_1,O_2) = <O_1,attr(O_1),O_2,attr(O_2),M>}$$

- 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 object class definitions or type definitions in a database.

- The objects enumerated above may model information at any level of representation (viz. 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 the object class 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 shall use the object algebra similar to the one defined in [SZ90] to express the mappings between O_1 and O_2. The object algebraic operations used have been defined in the Appendix A.2.

Schema Correspondences and Context

We consider structure to be a part of semantics. This is achieved by the association between the exported definition and association contexts and the object classes and types defined in the database. Each information system exports the definition contexts of the objects it manages. The exported context partially explicates the semantics of the object. This association might be implemented in different ways by various component systems. We use schema correspondences to express these associations.

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

- O_F is the exported federation object class of an object class or type definition O in the database.

- The attributes of the object O_F are the contextual coordinates of the definition context C_{def}(O).

- The rename operator $\mathbf{ren}_O\mathbf{(C_i,A_i)}$ stores the association between contextual coordinate C_i and attribute A_i of object class O whenever there exists one[13]

- The mapping M between O_F and O can be evaluated using the rules specified in the Appendix A.3 and illustrated in Section 7.2.

7.2 Schema Correspondences: Projection of semPro wrt Context

We discussed in Section 3 how representing structural similarities is not enough to capture semantic similarity between two object classes. However, for any meaningful operation to be performed on the computer, the semPro descriptor between two object classes has to be mapped to a mathematical expression which would essentially express the structural correspondence

[13]In some cases, a contextual coordinate C_i may not be associated with any attribute of the object class.

between two object classes. Our approach is to associate the schema correspondences discussed in the previous section with the context component of the semPro descriptor between two object classes.
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 it's first component, context. This includes intensional descriptions of:

- Object classes and their attributes.
- The relationships between various object classes.
- The implicit assumptions in the design of the object classes.
- The constraints which the object classes and attributes obey.

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 in Section 7.1 captures the mapping between the intensional descriptions and the object classes and types in a database. This is expressed by utilizing the object algebraic operations defined in the Appendix A.2. These correspondences retrieve objects from databases which satisfy the constraints specified in the context.

The mapping aspect can be succinctly expressed as:

$$\mathbf{schCor(O_1,O_2)} = \Pi_{Context}(\mathbf{semPro(O_1,O_2)})$$

Since the schema correspondences are associated with the context component of the semantic proximity, the schCor descriptor between two object classes is defined as the **Projection** of the semPro descriptor wrt the **Context**. In Figure 11, we have illustrated the mapping perspective. E1, E2, E3, E4 are entities in the real world and O1, O2, O3, O4 are their representations in the model word. Their relationships in the semantic space are represented by the semantic proximity descriptor and schema correspondences in the structural space. The semantic proximity is projected *wrt* the contexts C1, C2, C3 to give the schema correspondences.

We have defined an algebra of operations which help define the semantics of this Projection operation stated above. The algebra is presented in the Appendix A.3. A similar work on mapping intensional descriptions in CLASSIC to SQL queries has been done in [BB93]. In our approach however, the mappings to the actual data organization (which are expressed using object algebraic operations defined in the Appendix A.2) are also associated with the intensional descriptions (context which may be expressed using a CLASSIC like description logic language). Whenever the context changes, we also keep track of the associated changes in the schema correspondences. An algebra for the various changes in the schema correspondences is presented in the Appendix A.4 and illustrated with examples in Section 8.

In the rest of this section, we explain the relationship between the context and the schema correspondences with the help of examples. We shall use the terminology and operations defined in Appendix A.1, the object algebraic operations defined in Appendix A.2 and the rules defined for manipulating the semPro and schCor descriptors in Appendix A.3.

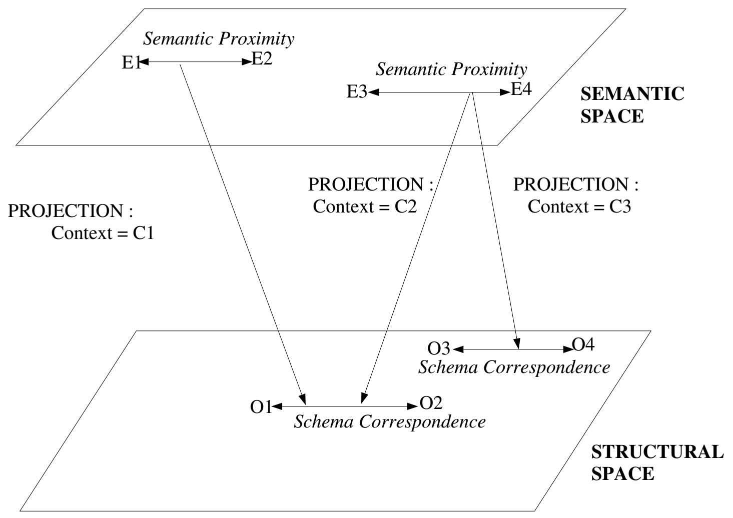


Figure 11: From the Semantic to the Structural: Projection wrt Context

We also demonstrate the *extra information* being retrieved by associating the intensional descriptions with the actual data organization. Thus, a context may be used to represent aspects of semantics not represented in the database.

7.2.1 Using ontology for an intensional description of data

In Section 4.1, we choose the contextual coordinates C_is and their values V_is from an ontology. We illustrate with the help of an example how information in an ontology may be mapped to the actual data in the database.

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

The definition context of the object class EMPLOYEE may be defined as:
C_{def}(EMPLOYEE) = <(employer,[**Deptypes**$\cup$\{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 $ren_{EMPLOYEE}$(C, A) operator.
- 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 7.1, we associate with definition context an object class $EMPLOYEE_F$ which is exported to the federation of databases.

$$\boxed{\begin{array}{l}\text{semPro}(\text{EMPLOYEE}_F,\text{EMPLOYEE}) \\ = \langle C_{def}(\text{EMPLOYEE}),\text{M},(\text{dom}(\text{EMPLOYEE}_F),\text{dom}(\text{EMPLOYEE})),\_\rangle\end{array}}$$

where M is a mapping between the domains of the two object classes. The mapping M is defined as a projection of the semPro descriptor and can be computed by the projection of the semPro descriptor *wrt* to the definition context. We use the rules defined in the Appendix A.3 to compute this mapping. It thus relates information in the ontology to data in the database. This projection is illustrated in Figure 12.

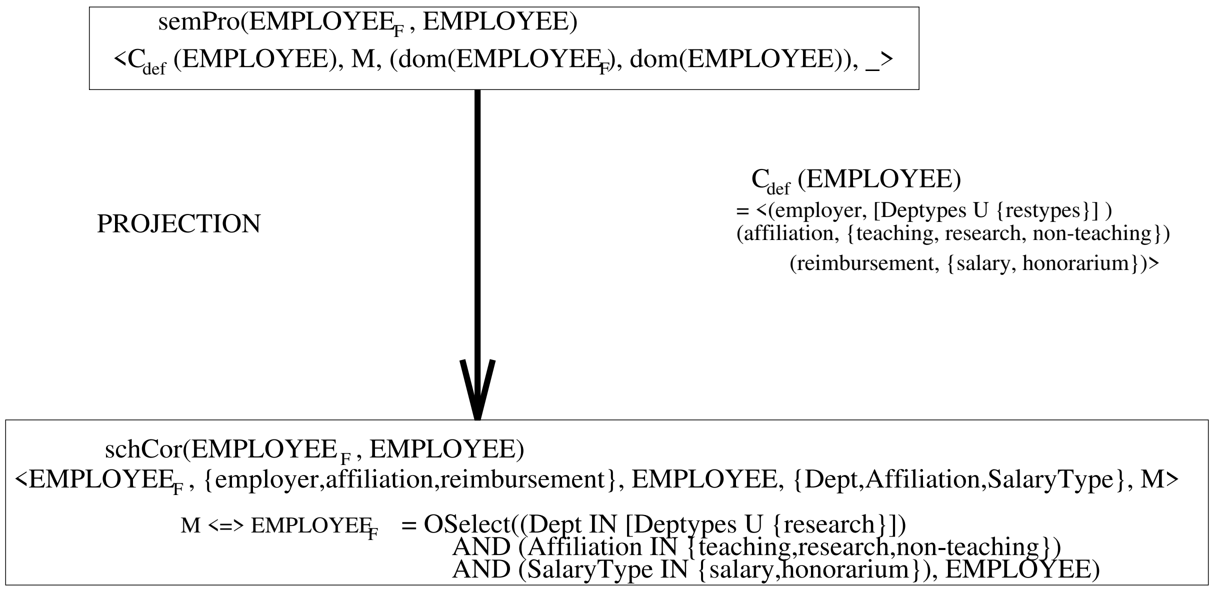

semPro(EMPLOYEE_F, EMPLOYEE)
<C_def (EMPLOYEE), M, (dom(EMPLOYEE_F), dom(EMPLOYEE)), _>
PROJECTION
C_def (EMPLOYEE)
= <(employer, [Deptypes U {restypes}])
(affiliation, {teaching, research, non-teaching})
(reimbursement, {salary, honorarium})>
schCor(EMPLOYEE_F, EMPLOYEE)
<EMPLOYEE_F, {employer,affiliation,reimbursement}, EMPLOYEE, {Dept,Affiliation,SalaryType}, M>
M <=> EMPLOYEE_F = OSelect((Dept IN [Deptypes U {research}])
AND (Affiliation IN {teaching,research,non-teaching})
AND (SalaryType IN {salary,honorarium}), EMPLOYEE)

Figure 12: Mapping EMPLOYEE_F to object class EMPLOYEE in the database

Simple Sets Projection (Rule 2) $\Rightarrow$

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

7.2.2 Domain Augmentation: Representing Extra Information

In this section, we demonstrate an interesting case where *extra information* can be stored with the intensional descriptions of object classes. This case arises when a contextual coordinate used to model the definition context of a class is not mapped to any of its attributes. We have defined a special case of the *strConstrain* operator (see Appendix A.3, Constraint Application, New Constraint, Non-existent Attribute Case, Rule 3.3) which would associate that information with all members of that object class. This operator provides the extra information represented which is not stored in the database. The extra information may however be used to perform inferences at the federation level.

Example:
Consider an object class PUBLICATION defined in the database as follows:
PUBLICATION(Id,Title,Journal)

The definition context of the object class PUBLICATION is defined as:
$C_{def}(\text{PUBLICATION}) = <(\text{researchArea},\textbf{Deptypes})>$

- **Deptypes** is a type defined in the database.
- The contextual coordinate researchArea is not associated with any of the attributes of the PUBLICATION, i.e. it is renamed to itself, $\text{ren}_{PUBLICATION}$(researchArea, researchArea).

Using the various rules from the algebra defined in the Appendix A.3, we now illustrate how extra information is represented and how the relevant mappings are computed. This is diagrammatically illustrated in Figure 13.

1. The semPro descriptor between PUBLICATION_F and PUBLICATION is:

   semPro(PUBLICATION_F,PUBLICATION)
   = <C_{def}(PUBLICATION),M,(dom(PUBLICATION_F),unknown),_>
   where M is a mapping between the domains listed in the semPro descriptor.
   C_{def}(PUBLICATION) = glb(<(researchArea,**Deptypes**)>,<>)

2. Item 1 and Constraint Application (Rule 3) $\Rightarrow$

   semPro(PUBLICATION_F,PUBLICATION)
   = semConstrain(<(researchArea,**Deptypes**)>,semPro(PUBLICATION_1,PUBLICATION)),
   where semPro(PUBLICATION_1,PUBLICATION)
   = <<>,M_1,(dom(PUBLICATION_1),dom(PUBLICATION)),_>

3. Item 2 and Empty Context Projection (Rule 1) $\Rightarrow$

   schCor(PUBLICATION_1,PUBLICATION) = <PUBLICATION_1,ϕ,PUBLICATION,ϕ,M_1 >
   $M_1 \equiv \text{PUBLICATION}_1$=PUBLICATION

4. Item 2,3 and Constraint Application (New Constraint, Non-existent Attribute, Rule 3.3) $\Rightarrow$

   schCor(PUBLICATION_F,PUBLICATION)
   = $\Pi_{C_{def}(PUBLICATION)}$(semConstrain(<(researchArea,**Deptypes**)>,
   semPro(PUBLICATION_1,PUBLICATION)))
   = strConstrain({$\text{ren}_{PUBLICATION}$(researchArea,researchArea)},**Deptypes**,
   schCor(PUBLICATION_1,PUBLICATION))
   = <PUBLICATION_F,{researchArea},PUBLICATION,
   {$\text{ren}_{PUBLICATION}$(researchArea,researchArea)},M>
   M $\equiv$ PUBLICATION_F=OProduct(makeObjectClass(researchArea,**Deptypes**),PUBLICATION_1)
   =OProduct(makeObjectClass(researchArea,**Deptypes**),PUBLICATION)

5. Item 4 $\Rightarrow$

   semPro(PUBLICATION_F,PUBLICATION)
   = <C_{def}(PUBLICATION),M,(dom(PUBLICATION_F),dom(PUBLICATION)$\times$**Deptypes**),_>

- The constraint of all publications having research areas which are associated with the Departments is not represented in the database.
- It is however represented in the object class PUBLICATION_F, which is exported to the federation. This is achieved by augmentation of dom(PUBLICATION), i.e., dom(PUBLICATION_F) $\subseteq$ dom(Id)$\times$dom(Title)$\times$dom(Journal)$\times$**Deptypes**.

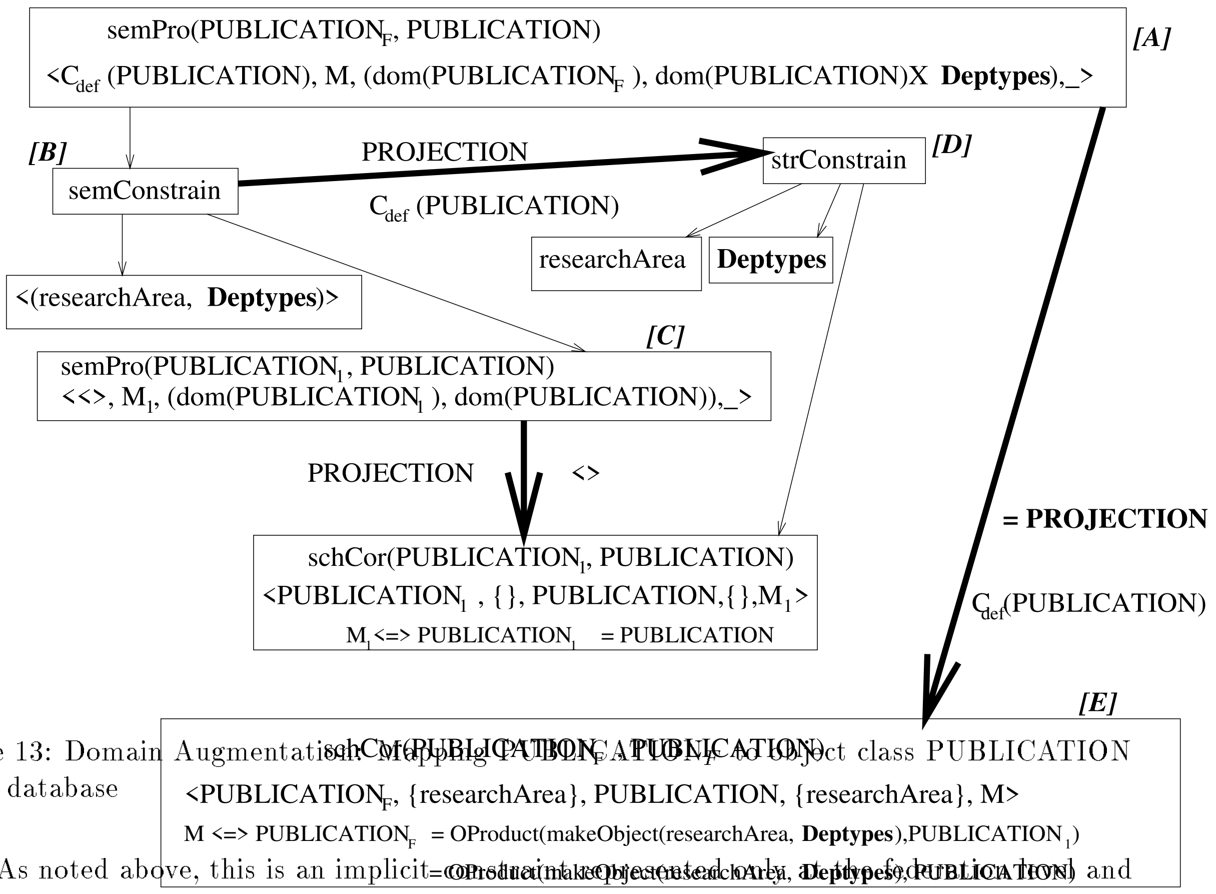

semPro(PUBLICATION_F, PUBLICATION)
<C_def (PUBLICATION), M, (dom(PUBLICATION_F), dom(PUBLICATION)X Deptypes),_>
[A]
[B]
semConstrain
PROJECTION
C_def (PUBLICATION)
strConstrain
[D]
researchArea
Deptypes
<(researchArea, Deptypes)>
[C]
semPro(PUBLICATION_1, PUBLICATION)
<<>, M_1, (dom(PUBLICATION_1), dom(PUBLICATION)),_>
PROJECTION
<>
= PROJECTION
C_def(PUBLICATION)
schCor(PUBLICATION_1, PUBLICATION)
<PUBLICATION_1 , {}, PUBLICATION,{},M_1>
M_1 <=> PUBLICATION_1 = PUBLICATION
[E]
schCor(PUBLICATION_F, PUBLICATION)
<PUBLICATION_F, {researchArea}, PUBLICATION, {researchArea}, M>
M <=> PUBLICATION_F = OProduct(makeObject(researchArea, Deptypes),PUBLICATION_1)
= OProduct(makeObject(researchArea, Deptypes), PUBLICATION)

Figure 13: Domain Augmentation: Mapping PUBLICATION$_F$ to object class PUBLICATION in the database

- As noted above, this is an implicit constraint represented only at the federation level and used to make inferences about information in the database without actually accessing the database.
- In this manner one can associate extra information available in the ontology with object classes in the database. Obviously, the information which can be associated thus is bounded by the ontology.

7.2.3 Representing relationships between object classes

In this section, we illustrate with the help of an example how context can be used to capture relationships between object classes which may not be represented in the database.

Example:
In Sections 7.2.1 and 7.2.2 we have defined the object classes EMPLOYEE and PUBLICATION. Consider an object class in the same database which represents a relationship between employees and the publications they write,
HAS-PUBLICATION(SS#,Id)

Using various rules from the algebra defined in the Appendix A.3, we now illustrate how extra information may be represented and how the relevant mappings may be computed. This is diagrammatically illustrated in Figure 14.

1. The semPro descriptor between HAS-PUBLICATION$_F$ and HAS-PUBLICATION is:

semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION)
= <C$_{def}$(HAS-PUBLICATION),M,(dom(HAS-PUBLICATION$_F$),unknown),_>
where C$_{def}$(HAS-PUBLICATION) = <(article,PUBLICATION)
(author, EMPLOYEE∘C$_{ass}$(EMPLOYEE, HAS-PUBLICATION))>
where C$_{ass}$(EMPLOYEE, HAS-PUBLICATION) = <(affiliation, {research})>

2. The definition context can be re-written as:

C$_{def}$(HAS-PUBLICATION)
= glb(<(author,EMPLOYEE∘C$_{ass}$(EMPLOYEE,HAS-PUBLICATION))>,
glb(<(article,PUBLICATION)>,<>))

3. Item 2 and Constraint Application (Rule 3), semPro Combination (Rule 5.1) ⇒

semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION)
= semConstrain(<(author,EMPLOYEE∘ <C$_{ass}$(EMPLOYEE,HAS-PUBLICATION))>,
semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION))
= semCombine(author,semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION)
semCondition(C$_{ass}$(EMPLOYEE,HAS-PUBLICATION),semPro(EMPLOYEE$_F$,EMPLOYEE)))

4. Item 3 and Empty Context Lifting (Rule 4.1), Constraint Lifting Application (Rule 4.2) ⇒

semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION)
= semCombine(author,semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION),
semConstrain(<(affiliation,{research})>,semPro(EMPLOYEE$_F$,EMPLOYEE)))

5. Consider the semPro descriptor between HAS-PUBLICATION$_1$ and HAS-PUBLICATION

semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION)
= <Cntxt$_1$,M$_1$,(dom(HAS-PUBLICATION$_1$),unknown),_>
where Cntxt$_1$ = glb(<(article,PUBLICATION)>,<>)
and can be rewritten as glb(<(article,PUBLICATION∘ <>)>,<>)

6. Item 5 and Constraint Application (Rule 3), semPro Combination (Rule 5), Empty Context Lifting (Rule 4.1) ⇒

semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION)
= semConstrain(<(article,PUBLICATION∘ <>)>,
semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION))
= semCombine(article,semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION),
semCondition(<>,semPro(PUBLICATION$_F$,PUBLICATION)))
= semCombine(article,semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION),
semPro(PUBLICATION$_F$,PUBLICATION))

7. Consider the semPro descriptor between HAS-PUBLICATION$_2$ and HAS-PUBLICATION

$$\boxed{\begin{array}{l}
\text{semPro(HAS-PUBLICATION}_2\text{,HAS-PUBLICATION)} \\
= <\text{Cntxt}_2\text{,M}_2\text{,(dom(HAS-PUBLICATION}_2\text{),dom(HAS-PUBLICATION)),\_}> \\
= <<>\text{,M}_2\text{,(dom(HAS-PUBLICATION}_2\text{),dom(HAS-PUBLICATION)),\_}> \\
\text{Items 2,3,6} \Rightarrow \text{Cntxt}_2 = <>
\end{array}}$$

8. Item 7 and Empty Context Projection (Rule 1) $\Rightarrow$

$$\boxed{\begin{array}{l}
\text{schCor(HAS-PUBLICATION}_2\text{,HAS-PUBLICATION)} \\
= <\text{HAS-PUBLICATION}_2,\phi,\text{HAS-PUBLICATION},\phi,\text{M}_2 > \\
\text{M}_2 \equiv \text{HAS-PUBLICATION}_2 = \text{HAS-PUBLICATION}
\end{array}}$$

9. Consider the semPro descriptor between PUBLICATION$_F$ and PUBLICATION

$$\boxed{\begin{array}{l}
\text{semPro(PUBLICATION}_F\text{,PUBLICATION)} \\
= <\text{C}_{def}\text{(PUBLICATION),M}_3\text{,(dom(PUBLICATION}_F\text{),dom(PUBLICATION)),\_}> \\
= <<>\text{,M}_3\text{,(dom(PUBLICATION}_F\text{),dom(PUBLICATION)),\_}> \\
\text{Assumption} \Rightarrow \text{C}_{def}\text{(PUBLICATION)} = <>
\end{array}}$$

10. Item 9 and Empty Context Projection (Rule 1) $\Rightarrow$

$$\boxed{\begin{array}{l}
\text{schCor(PUBLICATION}_F\text{,PUBLICATION)} \\
= <\text{PUBLICATION}_F,\phi,\text{PUBLICATION},\phi,\text{M}_3 > \\
\text{M}_3 \equiv \text{PUBLICATION}_F = \text{PUBLICATION}
\end{array}}$$

11. Item 6,7,9 and semPro Combination (New Constraint and Existent Attributes, Rule 5.1) $\Rightarrow$

$$\boxed{\begin{array}{l}
\text{schCor(HAS-PUBLICATION}_1\text{,HAS-PUBLICATION)} \\
\Pi_{Cntxt_1}\text{(semPro(HAS-PUBLICATION}_1\text{,HAS-PUBLICATION))} \\
= \text{strCombine}(\{\text{ren}_{HAS-PUBLICATION}\text{(article,Id),ren}_{PUBLICATION}\text{(article,Id)}\}, \\
\qquad\qquad\qquad \Pi_{<>}\text{(semPro(HAS-PUBLICATION}_2\text{,HAS-PUBLICATION)),} \\
\qquad\qquad\qquad \Pi_{<>}\text{(semPro(PUBLICATION}_F\text{,PUBLICATION)))}
\end{array}}$$

12. Item 8,10,11 and Empty Context Lifting and Projection (Rule 4.1), Empty Context Projection (Rule 1) $\Rightarrow$

$$\boxed{\begin{array}{l}
\text{schCor(HAS-PUBLICATION}_1\text{,HAS-PUBLICATION)} \\
= \text{strCombine(\{Id,Id\},schCor(HAS-PUBLICATION}_2\text{,HAS-PUBLICATION),} \\
\qquad\qquad\qquad \text{schCor(PUBLICATION}_F\text{,PUBLICATION))} \\
\text{semPro Combination (New Constraint and Existent Attributes, Rule 5.1)} \Rightarrow \\
= <\text{HAS-PUBLICATION}_1\text{,\{article\},\{HAS-PUBLICATION,PUBLICATION\},\{Id\},M}_4 > \\
\text{M}_4 \equiv \text{HAS-PUBLICATION}_1\text{=OJoin((Id=Id),HAS-PUBLICATION}_2\text{,PUBLICATION}_F\text{)} \\
\text{Item 10,11} \Rightarrow \\
\qquad\qquad\qquad\qquad \text{=OJoin((Id=Id),HAS-PUBLICATION,PUBLICATION)}
\end{array}}$$

13. Item 12 $\Rightarrow$

$$\boxed{\begin{array}{l}
\text{semPro(HAS-PUBLICATION}_1\text{,HAS-PUBLICATION)} \\
= <\text{Cntxt}_1\text{,M}_4\text{,} \\
\qquad \text{(dom(HAS-PUBLICATION}_1\text{),dom(HAS-PUBLICATION)}\times\text{dom(PUBLICATION)),\_}>
\end{array}}$$

14. Item 4,5 and semPro Combination (New Constraint and Existent Attributes, Rule 5.1) $\Rightarrow$

$$\begin{aligned}
&\mathrm{schCor(HAS\text{-}PUBLICATION}_F\mathrm{,HAS\text{-}PUBLICATION)}\\
&= \Pi_{C_{def}(HAS-PUBLICATION)}(\mathrm{semPro(HAS\text{-}PUBLICATION}_F\mathrm{,HAS\text{-}PUBLICATION))}\\
&= \mathrm{strCombine}(\{\mathrm{ren}_{EMPLOYEE}(\mathrm{author,SS\#}),\mathrm{ren}_{HAS-PUBLICATION}(\mathrm{author,SS\#})\},\\
&\qquad\qquad \Pi_{Cntxt_1}(\mathrm{semPro(HAS\text{-}PUBLICATION}_1\mathrm{,HAS\text{-}PUBLICATION)}),\\
&\qquad \Pi_{C_{ass}(EMPLOYEE,HAS-PUBLICATION)}(\mathrm{semConstrain}(<(\mathrm{affiliation},\{\mathrm{research}\})>,\\
&\qquad\qquad\qquad \mathrm{semPro(EMPLOYEE}_F\mathrm{,EMPLOYEE)))}
\end{aligned}$$

15. Example in Section 7.2.1 $\Rightarrow$

$$\begin{aligned}
&\mathrm{schCor(EMPLOYEE}_F\mathrm{,EMPLOYEE)}\\
&= <\mathrm{EMPLOYEE}_F,\{\mathrm{employer,affiliation,reimbursement}\},\mathrm{EMPLOYEE},\\
&\qquad\qquad \{\mathrm{Dept,Affiliation,SalaryType}\},\mathrm{M}_5>\\
&\mathrm{M}_5 \equiv \mathrm{EMPLOYEE}_F = \mathrm{OSelect(p,EMPLOYEE)}\\
&\mathrm{p} \equiv (\mathrm{Dept}\in[\mathrm{Deptypes}\cup\{\mathrm{restypes}\})\wedge(\mathrm{Affiliation}\in\{\mathrm{teaching,non\text{-}teaching,research}\})\wedge\\
&\quad (\mathrm{SalaryType}\in\{\mathrm{salary,honorarium}\})
\end{aligned}$$

16. Item 15 and Constraint Application (Modified Constraint and Existent Attribute, Rule 3.2) $\Rightarrow$

$$\begin{aligned}
&\mathrm{schCor(EMPLOYEE}_1\mathrm{,EMPLOYEE)}\\
&= \Pi_{C_{ass}(EMPLOYEE,HAS-PUBLICATION)}(\mathrm{semConstrain}(<(\mathrm{affiliation},\{\mathrm{research}\})>,\\
&\qquad\qquad \mathrm{semPro(EMPLOYEE}_F\mathrm{,EMPLOYEE)))}\\
&= \mathrm{strConstrain}(\mathrm{ren}_{EMPLOYEE}(\mathrm{affiliation,Affiliation}),\{\mathrm{research}\},\\
&\qquad\qquad \mathrm{schCor(EMPLOYEE}_F\mathrm{,EMPLOYEE))}\\
&= <\mathrm{EMPLOYEE}_1,\{\mathrm{affiliation}\},\mathrm{EMPLOYEE},\{\mathrm{Affiliation}\},\mathrm{M}_6>\\
&\mathrm{M}_6 \equiv \mathrm{EMPLOYEE}_1=\mathrm{OSelect}((\mathrm{Affiliation}\in\{\mathrm{research}\})\wedge\mathrm{p,EMPLOYEE})
\end{aligned}$$

17. Item 11,14,16 $\Rightarrow$

$$\begin{aligned}
&\mathrm{schCor(HAS\text{-}PUBLICATION}_F\mathrm{,HAS\text{-}PUBLICATION)}\\
&= \mathrm{strCombine}(\{\mathrm{SS\#,SS\#}\},\mathrm{schCor(HAS\text{-}PUBLICATION}_1\mathrm{,HAS\text{-}PUBLICATION)},\\
&\qquad\qquad \mathrm{schCor(EMPLOYEE}_1\mathrm{,EMPLOYEE))}
\end{aligned}$$

18. Item 12,16,17 $\Rightarrow$

$$\begin{aligned}
&\mathrm{schCor(HAS\text{-}PUBLICATION}_F\mathrm{,HAS\text{-}PUBLICATION)}\\
&= <\mathrm{HAS\text{-}PUBLICATION}_F,\{\mathrm{author,article}\},\\
&\quad \{\mathrm{PUBLICATION,HAS\text{-}PUBLICATION,EMPLOYEE}\},\{\mathrm{SS\#,Id}\},\mathrm{M}>\\
&\mathrm{M} \equiv \mathrm{HAS\text{-}PUBLICATION}_F = \mathrm{OJoin((SS\#=SS\#),HAS\text{-}PUBLICATION}_1\mathrm{,EMPLOYEE}_1)\\
&\qquad = \mathrm{OJoin((SS\#=SS\#),OJoin((Id=Id),HAS\text{-}PUBLICATION,PUBLICATION)},\\
&\qquad\qquad \mathrm{OSelect}((\mathrm{Affiliation}\in\{\mathrm{research}\})\wedge\mathrm{p,EMPLOYEE}))
\end{aligned}$$

19. Item 17 $\Rightarrow$

$$\begin{aligned}
&\mathrm{semPro(HAS\text{-}PUBLICATION}_F\mathrm{,HAS\text{-}PUBLICATION)}\\
&=<\mathrm{C}_{def}(\mathrm{HAS\text{-}PUBLICATION}),\mathrm{M},(\mathrm{dom(HAS\text{-}PUBLICATION}_F),\\
&\qquad \mathrm{dom(HAS\text{-}PUBLICATION)}\times\mathrm{dom(PUBLICATION)}\times\mathrm{dom(EMPLOYEE)}),\_>
\end{aligned}$$

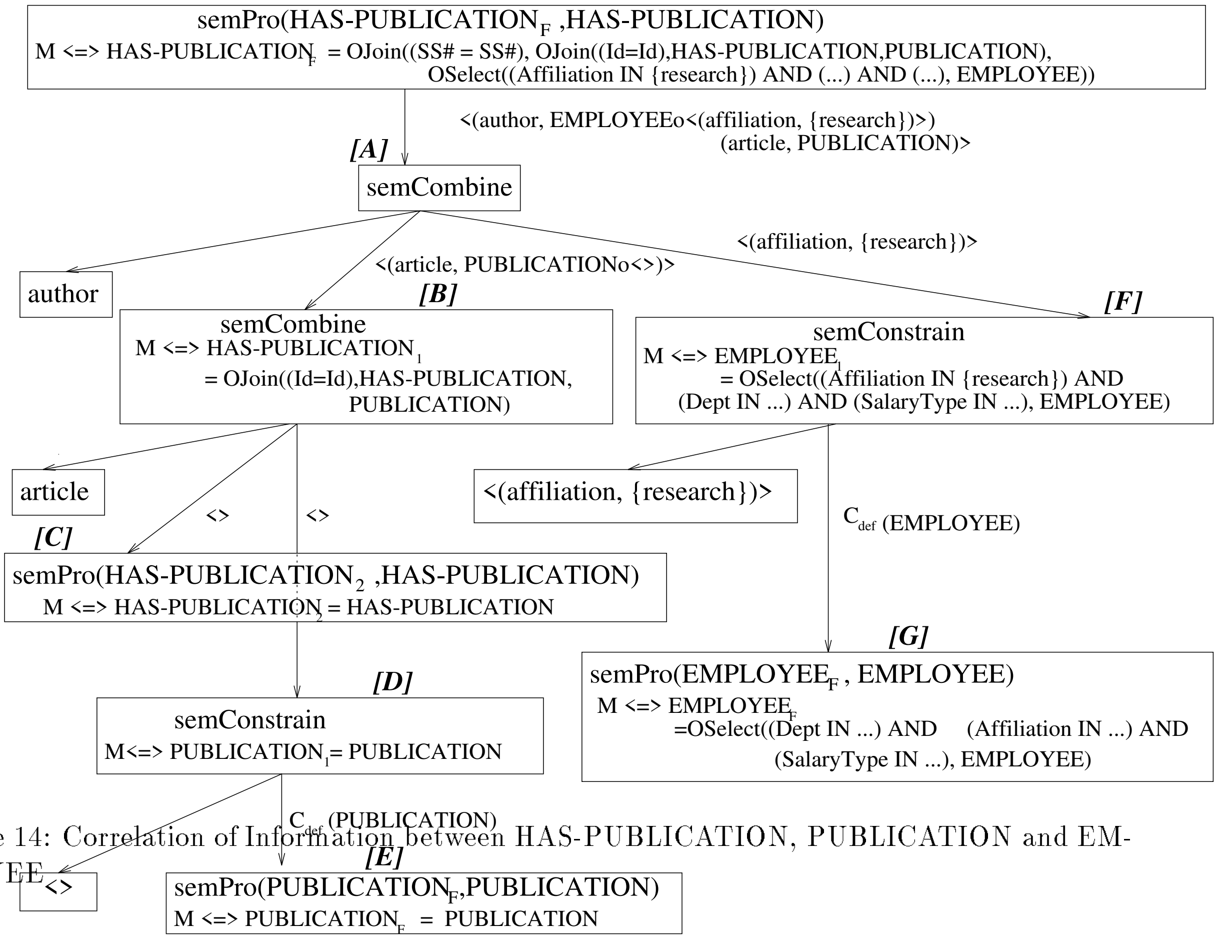

semPro(HAS-PUBLICATION_F ,HAS-PUBLICATION)
M <=> HAS-PUBLICATION_F = OJoin((SS# = SS#), OJoin((Id=Id),HAS-PUBLICATION,PUBLICATION), OSelect((Affiliation IN {research}) AND (...) AND (...), EMPLOYEE))
<(author, EMPLOYEEo<(affiliation, {research})>) (article, PUBLICATION)>
[A]
semCombine
author
<(article, PUBLICATIONo<>)>
<(affiliation, {research})>
[B]
semCombine
M <=> HAS-PUBLICATION_1 = OJoin((Id=Id),HAS-PUBLICATION, PUBLICATION)
[F]
semConstrain
M <=> EMPLOYEE_1 = OSelect((Affiliation IN {research}) AND (Dept IN ...) AND (SalaryType IN ...), EMPLOYEE)
article
<>
<>
<(affiliation, {research})>
C_def (EMPLOYEE)
[C]
semPro(HAS-PUBLICATION_2 ,HAS-PUBLICATION)
M <=> HAS-PUBLICATION_2 = HAS-PUBLICATION
[G]
semPro(EMPLOYEE_F , EMPLOYEE)
M <=> EMPLOYEE_F =OSelect((Dept IN ...) AND (Affiliation IN ...) AND (SalaryType IN ...), EMPLOYEE)
[D]
semConstrain
M<=> PUBLICATION_1= PUBLICATION
C_def (PUBLICATION)
[E]
<>
semPro(PUBLICATION_F,PUBLICATION)
M <=> PUBLICATION_F = PUBLICATION

Figure 14: Correlation of Information between HAS-PUBLICATION, PUBLICATION and EMPLOYEE

- In this case the object class HAS-PUBLICATION is defined in the context of the object classes EMPLOYEE and HAS-PUBLICATION already defined in the database. Thus the definition context of HAS-PUBLICATION depends on these two classes.
- A different perspective is that HAS-PUBLICATION represents a view capturing the semantic dependencies between the object classes and may not be actually stored in the database.
- Whenever HAS-PUBLICATION$_F$ is exported to the federation, it depends on EMPLOYEE$_F$ and PUBLICATION$_F$. This is reflected in the definition of the Projection operation.

7.2.4 Composition of Contextual Coordinates: Representing extra information

In this section, we illustrate an example in which extra information can be represented using intensional descriptions because of the following:

- One contextual coordinate is the composition[14] of contextual coordinates, each of which is mapped to the attributes of an object.
- One of the "subparts" of the contextual coordinate is not mapped to any of the attributes of another object (as in Section 7.2.2).

Example:
Consider a database containing the following object classes:
PUBLICATION(Id, Title, Journal)
JOURNAL(Title, Area)

Using the various rules from the algebra defined in the Appendix A.3, we now illustrate how extra information may be represented and how the relevant mappings may be computed. The resulting *correlation of information* is illustrated in Figure 15.

1. The semPro descriptor between PUBLICATION$_F$ and PUBLICATION is:

semPro(PUBLICATION$_F$,PUBLICATION) = <C$_{def}$(PUBLICATION),M,(dom(PUBLICATION$_F$),unknown),_> where C$_{def}$(PUBLICATION) = <(researchInfo,JOURNAL∘C$_{ass}$(JOURNAL,PUBLICATION))> and researchInfo = compose(researchArea,journalTitle) and C$_{ass}$(JOURNAL,PUBLICATION) = <(researchArea,Deptypes)(journalTitle,JournalTypes)>

2. The definition context can be re-written as:

C$_{def}$(PUBLICATION) = glb(<(researchInfo,JOURNAL∘C$_{ass}$(JOURNAL,PUBLICATION))>,<>)

3. Item 2 and Constraint Application (Rule 3), semPro Combination (Rule 5) ⇒

semPro(PUBLICATION$_F$,PUBLICATION)
= semConstrain(<(researchInfo,JOURNAL∘C$_{ass}$(JOURNAL,PUBLICATION))>,
semPro(PUBLICATION$_1$,PUBLICATION))
= semCombine(researchInfo,semPro(PUBLICATION$_1$,PUBLICATION),
semCondition(C$_{ass}$(JOURNAL,PUBLICATION),semPro(JOURNAL$_F$,JOURNAL)))

[14] May be obtained from the ontology.

4. Consider the semPro descriptor between $\mathrm{PUBLICATION}_1$, and PUBLICATION

$$
\boxed{\begin{array}{l}
\mathrm{semPro}(\mathrm{PUBLICATION}_1,\mathrm{PUBLICATION}) \\
= <\mathrm{Cntxt}_1,\mathrm{M}_1,(\mathrm{dom}(\mathrm{PUBLICATION}_1),\mathrm{dom}(\mathrm{PUBLICATION})),\_> \\
= <<>,\mathrm{M}_1,(\mathrm{dom}(\mathrm{PUBLICATION}_1),\mathrm{dom}(\mathrm{PUBLICATION})),\_> \\
\text{Item 2,3} \Rightarrow \mathrm{Cntxt}_1 = <>
\end{array}}
$$

5. Item 3 and Empty Context Projection (Rule 1) $\Rightarrow$

$$
\boxed{\begin{array}{l}
\mathrm{schCor}(\mathrm{PUBLICATION}_1,\mathrm{PUBLICATION}) \\
= <\mathrm{PUBLICATION}_1,\phi,\mathrm{PUBLICATION},\phi,\mathrm{M}_1 > \\
\mathrm{M}_1 \equiv \mathrm{PUBLICATION}_1 = \mathrm{PUBLICATION}
\end{array}}
$$

6. Constraint Lifting Application and Projection (Rule 4.2), Constraint Application (New Constraint and Existent Attributes (Rule 3.1) (each applied twice) and Empty Context Lifting and Projection (Rule 4.1), Empty Context Projection (Rule 1) $\Rightarrow$

$$
\boxed{\begin{array}{l}
\mathrm{schCor}(\mathrm{JOURNAL}_1,\mathrm{JOURNAL}) \\
= \Pi_{C_{ass}(JOURNAL,PUBLICATION)}(\mathrm{semCondition}(\mathrm{C}_{ass}(\mathrm{JOURNAL},\mathrm{PUBLICATION}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{semPro}(\mathrm{JOURNAL}_F,\mathrm{JOURNAL}))) \\
= <\mathrm{JOURNAL}_1,\{\mathrm{researchArea},\mathrm{journalTitle}\},\mathrm{JOURNAL}, \\
\qquad \{\mathrm{ren}_{JOURNAL}(\mathrm{researchArea},\mathrm{Area}),\mathrm{ren}_{JOURNAL}(\mathrm{journalTitle},\mathrm{Title})\},\mathrm{M}_2 > \\
\mathrm{M}_2 \equiv \mathrm{JOURNAL}_1 = \mathrm{OSelect}((\mathrm{Area} \in \mathrm{Deptypes}) \wedge (\mathrm{Title} \in \mathrm{JournalTypes}),\mathrm{JOURNAL})
\end{array}}
$$

7. Item 3 and semPro Combination Projection (Rule 5) $\Rightarrow$

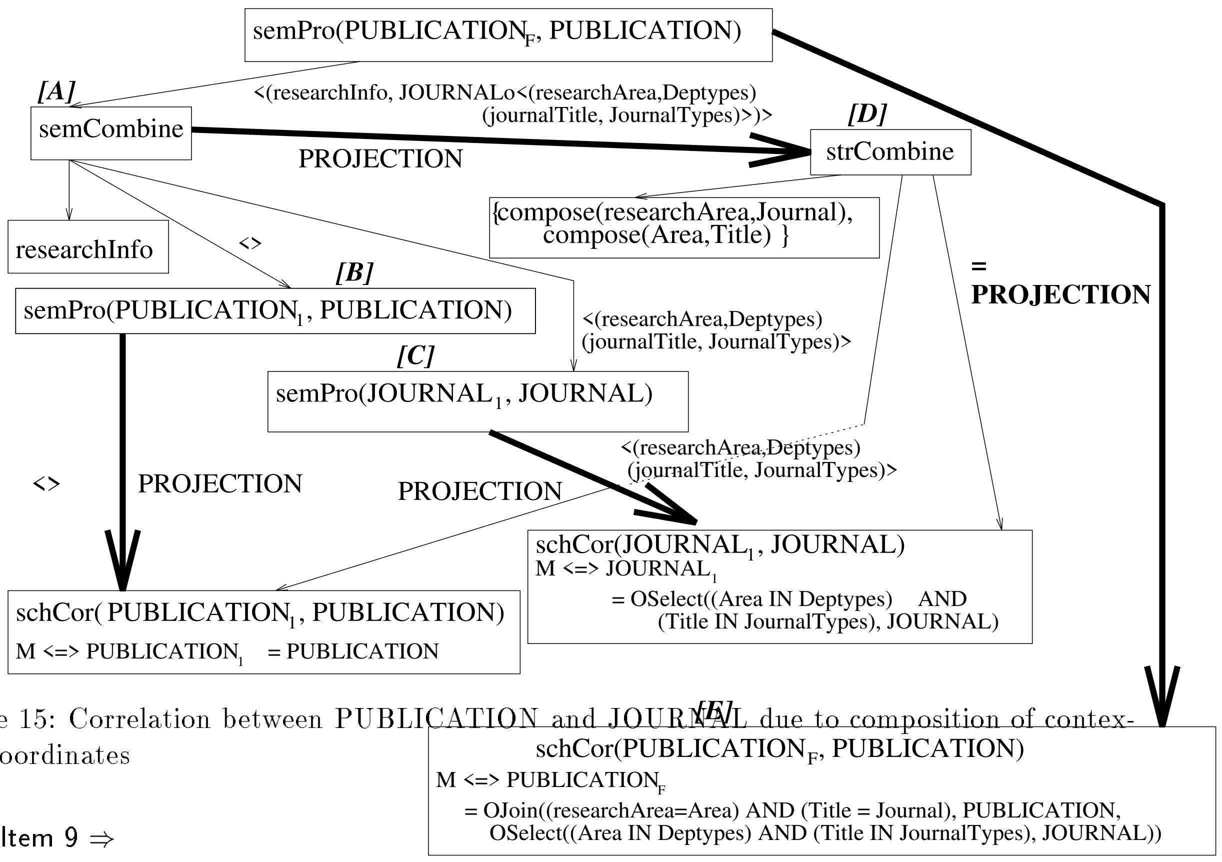
$$
\boxed{\begin{array}{l}
\mathrm{schCor}(\mathrm{PUBLICATION}_F,\mathrm{PUBLICATION}) \\
= \Pi_{C_{def}(PUBLICATION)}(\mathrm{semPro}(\mathrm{PUBLICATION}_F,\mathrm{PUBLICATION})) \\
= \mathrm{strCombine}(\{\mathrm{ren}_{PUBLICATION}(\mathrm{researchInfo},\mathrm{X}),\mathrm{ren}_{JOURNAL}(\mathrm{researchInfo},\mathrm{Y})\}, \\
\qquad\qquad\qquad\qquad\qquad \Pi_{Cntxt_1}(\mathrm{semPro}(\mathrm{PUBLICATION}_1,\mathrm{PUBLICATION})), \\
\qquad\qquad \Pi_{C_{ass}(JOURNAL,PUBLICATION)}(\mathrm{semCondition}(\mathrm{C}_{ass}(\mathrm{JOURNAL},\mathrm{PUBLICATION}), \\
\qquad\qquad\qquad\qquad\qquad \mathrm{semPro}(\mathrm{JOURNAL}_F,\mathrm{JOURNAL}))))
\end{array}}
$$

8. Item 1 and Contextual Coordinate Composition (Rule 5.3) $\Rightarrow$

$$
\boxed{\begin{array}{l}
\mathrm{ren}_{PUBLICATION}(\mathrm{researchInfo},\mathrm{X}) \\
= \mathrm{compose}(\mathrm{ren}_{PUBLICATION}(\mathrm{researchArea},\mathrm{researchArea}),\mathrm{ren}_{PUBLICATION}(\mathrm{journalTitle},\mathrm{Journal}) \\
\mathrm{ren}_{JOURNAL}(\mathrm{researchInfo},\mathrm{X}) \\
= \mathrm{compose}(\mathrm{ren}_{JOURNAL}(\mathrm{researchArea},\mathrm{Area}),\mathrm{ren}_{JOURNAL}(\mathrm{journalTitle},\mathrm{Title}))
\end{array}}
$$

9. Item 4,5,6,7,8 $\Rightarrow$

$$
\boxed{\begin{array}{l}
\mathrm{schCor}(\mathrm{PUBLICATION}_F,\mathrm{PUBLICATION}) \\
= \mathrm{strCombine}(\{\mathrm{compose}(\mathrm{researchArea},\mathrm{Journal}),\mathrm{compose}(\mathrm{Area},\mathrm{Title})\}, \\
\qquad\qquad \mathrm{schCor}(\mathrm{PUBLICATION}_1,\mathrm{PUBLICATION}),\mathrm{schCor}(\mathrm{JOURNAL}_1,\mathrm{JOURNAL})) \\
= <\mathrm{PUBLICATION}_F,\{\mathrm{researchInfo}\},\{\mathrm{JOURNAL},\mathrm{PUBLICATION}\}, \\
\qquad\qquad\qquad\qquad\qquad \{\mathrm{Title},\mathrm{Journal},\mathrm{Area},\mathrm{researchArea}\},\mathrm{M}> \\
\mathrm{M} \equiv \mathrm{PUBLICATION}_F = \mathrm{OJoin}((\mathrm{researchArea}=\mathrm{Area}) \wedge (\mathrm{Title}=\mathrm{Journal}),\mathrm{PUBLICATION}, \\
\qquad\qquad\qquad\qquad \mathrm{OSelect}((\mathrm{Area} \in \mathrm{Deptypes}) \wedge (\mathrm{Title} \in \mathrm{JournalTypes}),\mathrm{JOURNAL}))
\end{array}}
$$



semPro(PUBLICATION_F, PUBLICATION)
[A]
<(researchInfo, JOURNALo<(researchArea,Deptypes) (journalTitle, JournalTypes)>)>
[D]
semCombine
PROJECTION
strCombine
{compose(researchArea,Journal), compose(Area,Title) }
researchInfo
<>
[B]
=
PROJECTION
semPro(PUBLICATION_1, PUBLICATION)
<(researchArea,Deptypes) (journalTitle, JournalTypes)>
[C]
semPro(JOURNAL_1, JOURNAL)
<(researchArea,Deptypes) (journalTitle, JournalTypes)>
<>
PROJECTION
PROJECTION
schCor(JOURNAL_1, JOURNAL)
M <=> JOURNAL_1
= OSelect((Area IN Deptypes) AND (Title IN JournalTypes), JOURNAL)
schCor(PUBLICATION_1, PUBLICATION)
M <=> PUBLICATION_1 = PUBLICATION
[E]
schCor(PUBLICATION_F, PUBLICATION)
M <=> PUBLICATION_F
= OJoin((researchArea=Area) AND (Title = Journal), PUBLICATION, OSelect((Area IN Deptypes) AND (Title IN JournalTypes), JOURNAL))

Figure 15: Correlation between PUBLICATION and JOURNAL due to composition of contextual coordinates

10. Item 9 ⇒

$$\boxed{\begin{array}{l}\text{semPro(PUBLICATION}_F\text{,PUBLICATION)}\\ = \text{<C}_{def}\text{(PUBLICATION),M,(dom(PUBLICATION}_F\text{),}\\ \qquad\qquad\qquad\text{dom(PUBLICATION)}\times\text{dom(JOURNAL)),\_>}\end{array}}$$

Several things may be noted here:

- Similar to the example in Section 7.2.2, the contextual coordinate researchArea is not mapped to the attributes of the object class PUBLICATION.
- In this case, extra information is being added to the PUBLICATION$_F$ object exported to the federation. The composition of coordinates leads to a selective and implicit domain augmentation of Deptypes to the object class PUBLICATION, through the OJoin operation.
- This type of a relationship can not be expressed in the database as it depends on the semantic composition of contextual coordinates such that one of them is not mapped to attributes of an object class. Without the knowledge of composition of attributes, this would not be possible.

7.2.5 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 class PUBLICATION.

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

This represents a constraint on the instances of the object class 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 class PUBLICATION$_F$ at the federation level and can help in querying the database in face of incomplete information.

8 Applications of Context

In Section 7.2, we showed modeling of schema correspondences as the projection of the semPro descriptor wrt the definition context. In this section, we shall 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. The rules of the algebra defined in Appendix A.4 help us illustrate the changes in the schema correspondences as a result of the changes in the context they are associated with.

We show how query processing can be implemented by the comparison of the definition contexts of the object classes in the database with the query context. In particular, we illustrate the processes of *information focusing* and *information correlation.* We shall use the example in [KS94] to illustrate some key points. Throughout this section we shall use the following query and it's associated context:

Get all Congressmen and Senators who have published papers on the socio-political implications of the abortion issue.

C_Q = <(author,X) (designee,X) (employer,{legislative,restypes})
(article,Y$\circ$ <(title,{u| substring(u) = "abortion"})>)
(researchArea,{socialSciences,politics})>

8.1 Information Focusing: Modification of Schema Correspondence

In this section we illustrate how information focusing takes place as a result of comparing the query context and the definition context of an object class in the database. This is a simple case where the definition context of the object class doesn't depend on other object classes.

In Figure 16, we compare C_{def}(EMPLOYEE) and C_Q. This helps us identify all the employees who do research as being relevant to the query Q.

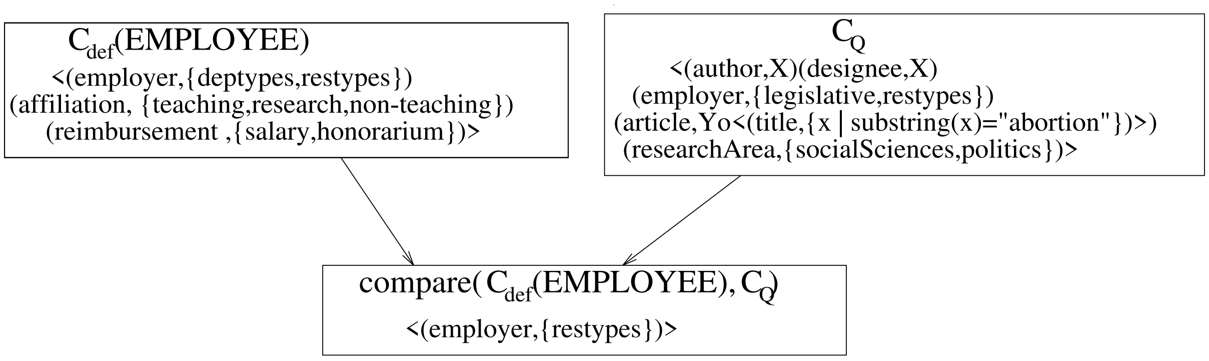

C_{def}(EMPLOYEE)
<(employer,{deptypes,restypes})
(affiliation, {teaching,research,non-teaching})
(reimbursement ,{salary,honorarium})>
C_Q
<(author,X)(designee,X)
(employer,{legislative,restypes})
(article,Yo<(title,{x | substring(x)="abortion"})>)
(researchArea,{socialSciences,politics})>
compare(C_{def}(EMPLOYEE), C_Q)
<(employer,{restypes})>

Figure 16: Context Comparison: Focusing on the relevant employees

Using the rules defined in the algebra in Appendix A.4, we illustrate how the schema correspondence associated with the definition context changes as a result of the changes in the definition context. The changes in the definition context are diagrammatically illustrated in Figure 17.

1. From Section 7.2.1 we have:

   schCor(EMPLOYEE$_F$,EMPLOYEE)
   = <EMPLOYEE$_F$,{employer,affiliation,reimbursement},EMPLOYEE,
   {Dept,Affiliation,SalaryType},M>
   M ≡ EMPLOYEE$_F$ = OSelect(p,EMPLOYEE)
   p ≡ (Dept∈[**Deptypes**∪{restypes}])∧(Affiliation∈{teaching,research,non-teaching})∧
   (SalaryType∈{salary,honorarium})

2. Specialized Constraint Lifting Application (Rule 7.2) ⇒

   semCompare(C_Q,semPro(EMPLOYEE$_F$,EMPLOYEE))
   = comCompare(<(author,X)>,semCompare(Cntxt$_1$,semPro(EMPLOYEE$_F$,EMPLOYEE)))
   where C_Q = glb(<(author,X)>,Cntxt$_1$)

3. Since author$\notin C_{def}$(EMPLOYEE) Item 2 and Focusing Constraints (New Constraint, Non-existent Attribute, Rule 6.1) ⇒

   semCompare(C_Q,semPro(EMPLOYEE$_F$,EMPLOYEE)
   = semCompare(Cntxt$_1$,semPro(EMPLOYEE$_F$,EMPLOYEE))

4. Repeated applications of Specialized Constraint Lifting Application (Rule 7.2) and Focusing Constraints (New Constraint, Non-existent Attribute, Rule 6.1) ⇒

semCompare(C_Q,semPro(EMPLOYEE$_F$,EMPLOYEE)
= comCompare(<(employer,{legislative,restypes}>,semPro(EMPLOYEE$_F$,EMPLOYEE))
Focusing Constraints (Modified Constraint, Existent attribute, Rule 6.2) and Constraint Application (Modified Constraint, Existent attribute, Rule 3.2) ⇒
= semConstrain(<(employer,{restypes})>,semPro(EMPLOYEE$_1$,EMPLOYEE))
where Cntxt = glb(<(employer,{restypes})>,Cntxt$_1$)
and Cntxt$_1$ = <(affiliation,{teaching,research,non-teaching})
(reimbursement,{salary,honorarium})>

5. Item 4 and two applications of Constraint Application (New Constraint and Existent Attribute, Rule 3.1) and Empty Context Projection (Rule 1) ⇒

schCor(EMPLOYEE$_1$,EMPLOYEE)
= Π_{Cntxt_1}(semPro(EMPLOYEE$_1$,EMPLOYEE))
= <EMPLOYEE$_1$,{affiliation,reimbursement},EMPLOYEE,{Affiliation,SalaryType},M$_1$ >
M$_1$ ≡ EMPLOYEE$_1$=OSelect(p,EMPLOYEE)
p ≡ (Affiliation∈{teaching,research,non-teaching})∧(SalaryType∈{salary,honorarium})

6. Item 1,4,5 and Constraint Application Projection (Rule 3) ⇒

Π_{Cntxt}(semConstrain(<(employer,{restypes})>,semPro(EMPLOYEE$_1$,EMPLOYEE)))
= strConstrain(ren$_{EMPLOYEE}$(employer,Dept),{restypes},
Π_{Cntxt_1}(semPro(EMPLOYEE$_1$,EMPLOYEE)))
= strConstrain(Dept,{restypes},schCor(EMPLOYEE$_1$,EMPLOYEE))
= <ANSWER,{employer,affiliation,reimbursement},EMPLOYEE,
{Dept,Affiliation,SalaryType},M$_1$ >
M$_1$ ≡ ANSWER=OSelect((Dept∈{restypes}),EMPLOYEE$_1$)
=OSelect((Dept∈{restypes})∧p,EMPLOYEE)

Thus we see how the schema correspondence between EMPLOYEE$_F$ and EMPLOYEE is modified to focus onto the relevant employees.

8.2 Information Focusing: Modification of Schema Correspondence of a related object class

In this section we illustrate how information focusing takes place by modifying the schema correspondence of a related object class. In this case, the comparison takes place between the query context and the definition context of an object class which contains another object class.

In Figure 18 we compare C_{def}(PUBLICATION) with C_Q and determine the research areas social sciences and politics as being relevant to the query Q.

Using the rules defined in the algebra in Appendix A.4, we illustrate how the schema correspondence associated with the definition context changes as a result of the changes in the definition context. The changes in the definition context are diagrammatically illustrated in Figure 19.

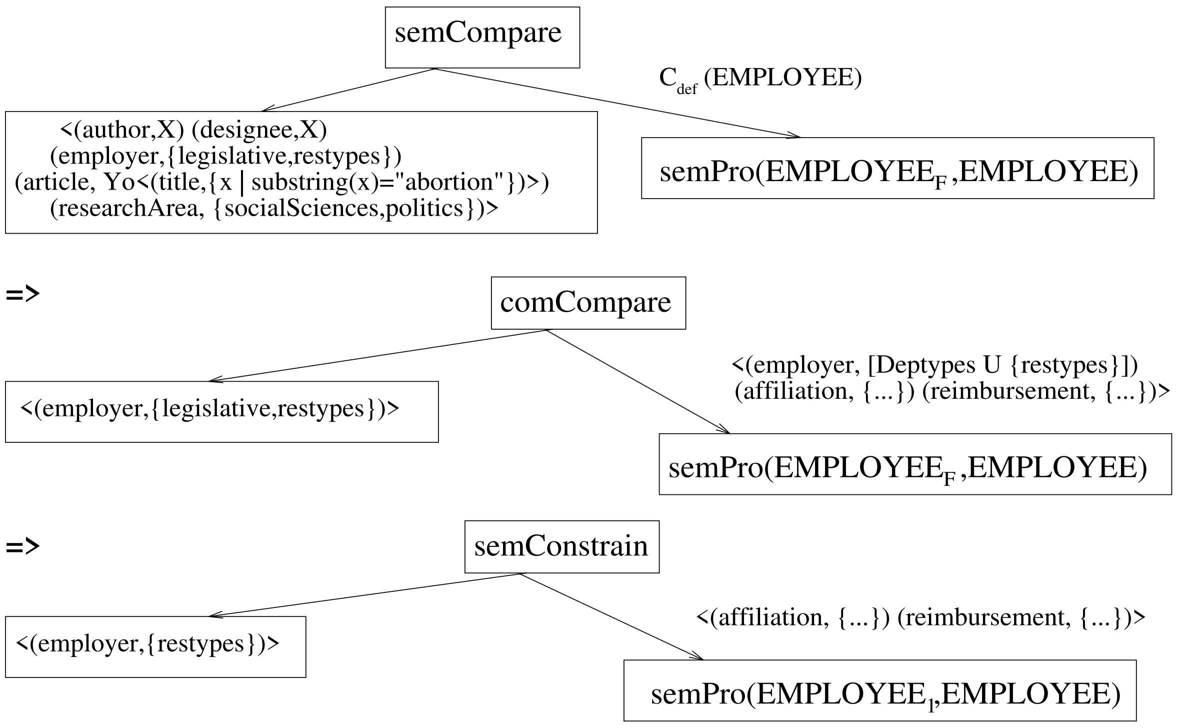

semCompare
C_{def} (EMPLOYEE)
<(author,X) (designee,X) (employer,{legislative,restypes}) (article, Yo<(title,{x | substring(x)="abortion"})>) (researchArea, {socialSciences,politics})>
semPro(EMPLOYEE$_F$,EMPLOYEE)
=>
comCompare
<(employer,{legislative,restypes})>
<(employer, [Deptypes U {restypes}]) (affiliation, {...}) (reimbursement, {...})>
semPro(EMPLOYEE$_F$,EMPLOYEE)
=>
semConstrain
<(employer,{restypes})>
<(affiliation, {...}) (reimbursement, {...})>
semPro(EMPLOYEE$_I$,EMPLOYEE)

Figure 17: Information Focusing: The main steps

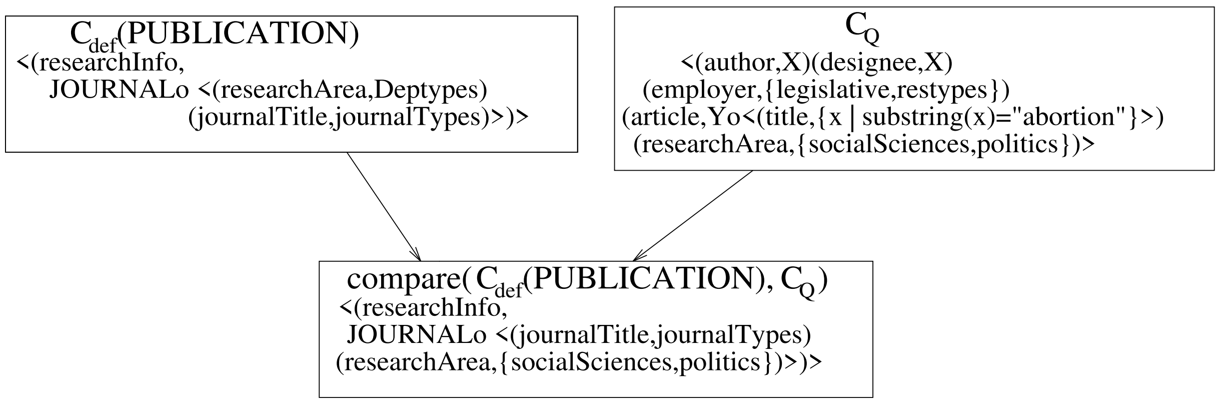

C_{def}(PUBLICATION)
<(researchInfo, JOURNALo <(researchArea,Deptypes) (journalTitle,journalTypes)>)>
C_Q
<(author,X)(designee,X) (employer,{legislative,restypes}) (article,Yo<(title,{x | substring(x)="abortion"}>) (researchArea,{socialSciences,politics})>
compare(C_{def}(PUBLICATION), C_Q)
<(researchInfo, JOURNALo <(journalTitle,journalTypes) (researchArea,{socialSciences,politics})>)>

Figure 18: Context Comparison: Focusing on the relevant research areas

1. From the example in Section 7.2.4 we have:

   $$\boxed{\begin{array}{l} \text{semPro}(\text{PUBLICATION}_F,\text{PUBLICATION}) \\ = \text{semCombine}(\text{researchInfo},\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION}), \\ \qquad \text{semCondition}(\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION}),\text{semPro}(\text{JOURNAL}_F,\text{JOURNAL}))) \\ \text{C}_{def}(\text{PUBLICATION}) = <(\text{researchInfo},\text{JOURNAL}\circ\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION}))> \\ \text{schCor}(\text{PUBLICATION}_F,\text{PUBLICATION}) \\ = <\text{PUBLICATION}_F,\{\text{researchInfo}\},\{\text{PUBLICATION},\text{JOURNAL}\}, \\ \qquad\qquad\qquad\qquad \{\text{Title},\text{Journal},\text{Area},\text{researchArea}\},\text{M}_1 > \\ \text{M}_1 \equiv \text{PUBLICATION}_F = \text{OJoin}((\text{researchArea}=\text{Area})\wedge(\text{Title}=\text{Journal}),\text{PUBLICATION}, \\ \qquad\qquad \text{OSelect}((\text{Area}\in\text{Deptypes})\wedge(\text{Title}\in\text{JournalTypes}),\text{JOURNAL})) \end{array}}$$

2. Item 1 and Specialized Context Lifting and semPro Combination (Rule 7.3) $\Rightarrow$

   $$\boxed{\begin{array}{l} \text{semCompare}(\text{C}_Q,\text{semPro}(\text{PUBLICATION}_F,\text{PUBLICATION})) \\ = \text{semCompare}(\text{C}_Q,\text{semCombine}(\text{researchInfo},\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION}), \\ \qquad \text{semCondition}(\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION}),\text{semPro}(\text{JOURNAL}_F,\text{JOURNAL})))) \\ = \text{semCombine}(\text{researchInfo},\text{semCompare}(\text{C}_Q,\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION})), \\ \qquad\qquad \text{semCompare}(\text{C}_Q,\text{semCondition}(\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION}), \\ \qquad\qquad\qquad\qquad \text{semPro}(\text{JOURNAL}_F,\text{JOURNAL})))) \end{array}}$$

3. Repeated Applications of Focusing Constraints (New Constraints, Non-existent Attribute, Rule 6.1) and Specialized Constraint Lifting Application (Rule 7.5) $\Rightarrow$

   $$\boxed{\begin{array}{l} \text{semCompare}(\text{C}_Q,\text{semPro}(\text{PUBLICATION}_F,\text{PUBLICATION})) \\ = \text{semCombine}(\text{researchInfo},\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION}), \\ \qquad\qquad \text{semCompare}(\text{C}_Q,\text{semCondition}(\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION}), \\ \qquad\qquad\qquad\qquad \text{semPro}(\text{JOURNAL}_F,\text{JOURNAL})))) \end{array}}$$

4. Item 3 and Specialized and Ordinary Context Lifting (Rule 7.4) $\Rightarrow$

   $$\boxed{\begin{array}{l} \text{semCompare}(\text{C}_Q,\text{semPro}(\text{PUBLICATION}_F,\text{PUBLICATION})) \\ = \text{semCombine}(\text{researchInfo},\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION}), \\ \quad \text{semCompare}(\text{glb}(\text{C}_Q,\text{C}_{ass}(\text{JOURNAL},\text{PUBLICATION})),\text{semPro}(\text{JOURNAL}_F,\text{JOURNAL}))) \\ = \text{semCombine}(\text{researchInfo},\text{semPro}(\text{PUBLICATION}_1,\text{PUBLICATION}), \\ \qquad\qquad \text{semCompare}(<(\text{researchArea},\{\text{socialSciences},\text{politics}\})(\text{journalTitle},\text{journalTypes})>, \\ \qquad\qquad\qquad\qquad \text{semPro}(\text{JOURNAL}_F,\text{JOURNAL}))) \end{array}}$$

5. Thus we see from the 3rd argument of the semCombine operator, that though the definition context of PUBLICATION and the query context are compared, it is the schema correspondence between JOURNAL$_F$ and JOURNAL which is modified to find the relevant publications.

6. Considering the relevant projection operations we have:

   $$\boxed{\begin{array}{l} \text{schCor}(\text{JOURNAL}_2,\text{JOURNAL}) \\ = \Pi_{<(researchArea,\{socialSciences,politics\})(journalTitle,journalTypes)>}(\text{semPro}(\text{JOURNAL}_2,\text{JOURNAL})) \\ = <(\text{JOURNAL}_2,\{\text{researchArea},\text{journalTitle}\},\text{JOURNAL},\{\text{Area},\text{Title}\},\text{M}_1 > \\ \text{M}_1 \equiv \text{JOURNAL}_2 = \text{OSelect}((\text{Area}\in\{\text{socialSciences},\text{politics}\})\wedge(\text{Title}\in\text{JournalTypes}),\text{JOURNAL}) \end{array}}$$

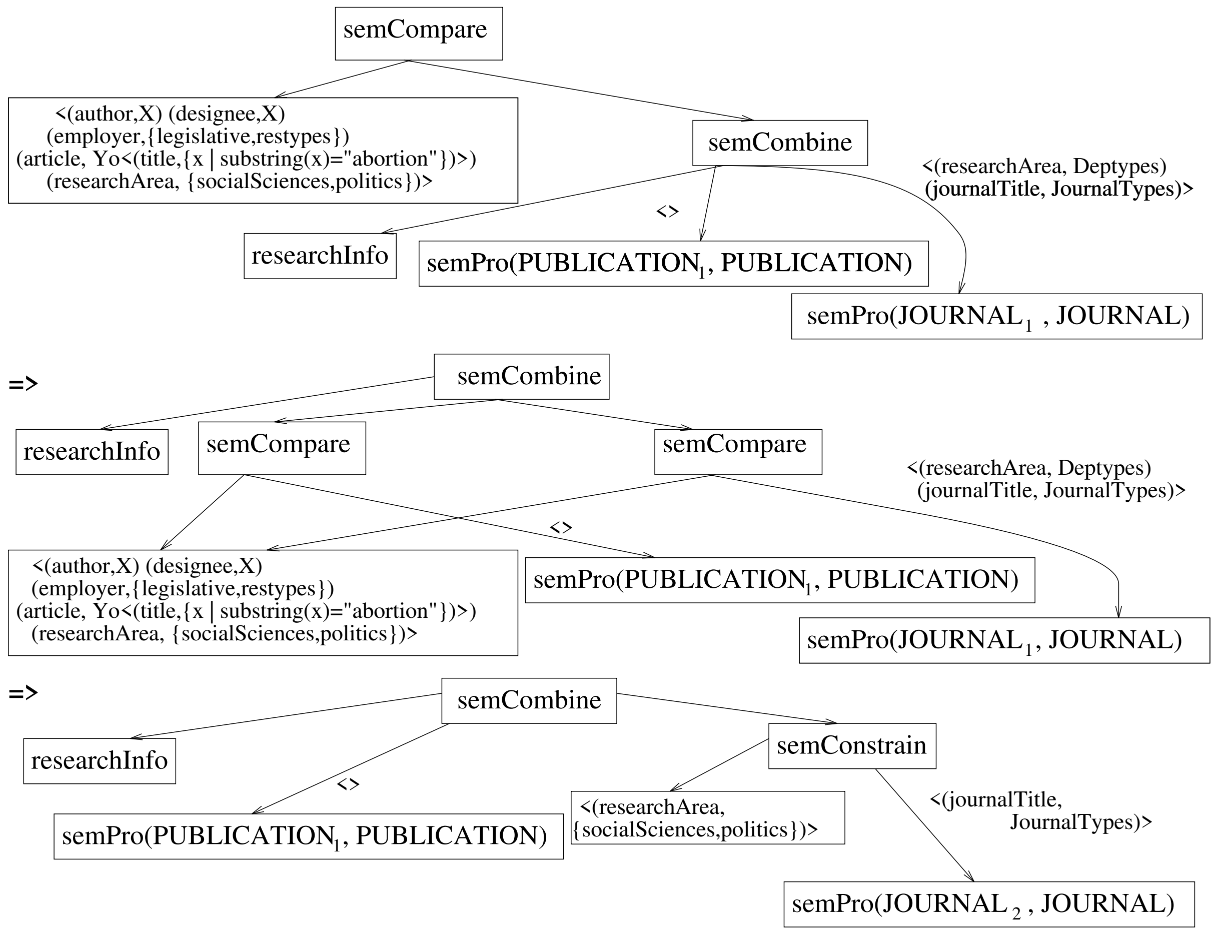

semCompare
<(author,X) (designee,X) (employer,{legislative,restypes}) (article, Yo<(title,{x | substring(x)="abortion"})>) (researchArea, {socialSciences,politics})>
semCombine
<(researchArea, Deptypes) (journalTitle, JournalTypes)>
<>
researchInfo
semPro(PUBLICATION1, PUBLICATION)
semPro(JOURNAL1 , JOURNAL)
=>
semCombine
researchInfo
semCompare
semCompare
<(researchArea, Deptypes) (journalTitle, JournalTypes)>
<>
<(author,X) (designee,X) (employer,{legislative,restypes}) (article, Yo<(title,{x | substring(x)="abortion"})>) (researchArea, {socialSciences,politics})>
semPro(PUBLICATION1, PUBLICATION)
semPro(JOURNAL1, JOURNAL)
=>
semCombine
researchInfo
semConstrain
<>
<(researchArea, {socialSciences,politics})>
<(journalTitle, JournalTypes)>
semPro(PUBLICATION1, PUBLICATION)
semPro(JOURNAL2 , JOURNAL)

Figure 19: Propagation of Information Focusing from JOURNAL to PUBLICATION

7. From the example in Section 7.2.4 we have:

$$\boxed{\begin{array}{l}\text{schCor(PUBLICATION}_1\text{,PUBLICATION)}\\ = <\text{PUBLICATION}_1,\phi,\text{PUBLICATION},\phi,\text{M}_2 >\\ \text{M}_2 \equiv \text{PUBLICATION}_1\text{=PUBLICATION}\end{array}}$$

8. Item 6,7 and applying the relevant projection rules $\Rightarrow$

$$\boxed{\begin{array}{l}\text{strCombine(\{ren}_{PUBLICATION}\text{(researchInfo,X),ren}_{JOURNAL}\text{(researchInfo,Y)\},}\\ \qquad\qquad\text{schCor(PUBLICATION}_1\text{,PUBLICATION),schCor(JOURNAL}_2\text{,JOURNAL))}\\ = <\text{ANSWER,\{researchInfo\},\{PUBLICATION,JOURNAL\},}\\ \qquad\qquad\qquad\qquad\text{\{Title,Journal,Area,researchArea\},M}_3 >\\ \text{M}_3 \equiv \text{ANSWER} = \text{OJoin((researchArea=Area)}\wedge\text{(Title=Journal),PUBLICATION}_1\text{,JOURNAL}_2\text{))}\\ \qquad\qquad = \text{OJoin((researchArea=Area)}\wedge\text{(Title=Journal),PUBLICATION,}\\ \qquad\qquad\qquad \text{OSelect((Area}\in\text{\{socialSciences,politics\})}\wedge\text{(Title}\in\text{JournalTypes),JOURNAL))}\end{array}}$$

There are various types of information focusing taking place here:

- In the JOURNAL object class only the instances that belong to the areas of the socialSciences and politics are considered. This helps to focus on the journals belonging to only those two areas.

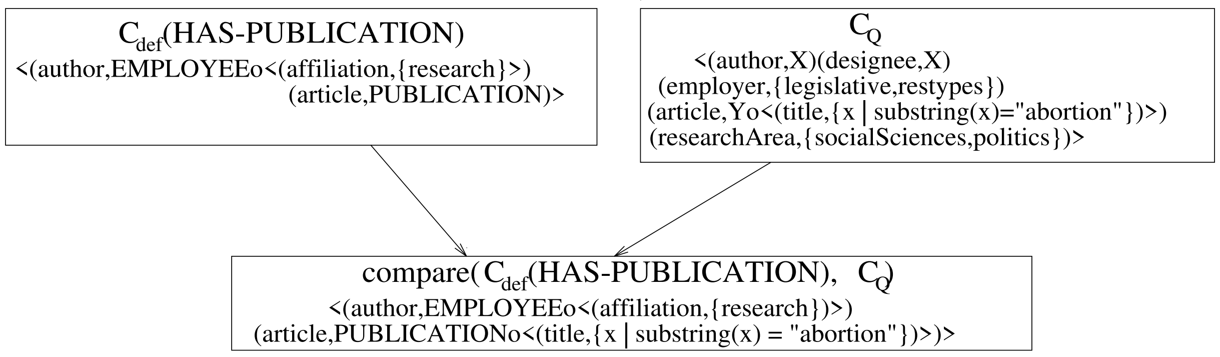

C_def(HAS-PUBLICATION)
<(author,EMPLOYEEo<(affiliation,{research}>)
(article,PUBLICATION)>
C_Q
<(author,X)(designee,X)
(employer,{legislative,restypes})
(article,Yo<(title,{x | substring(x)="abortion"})>)
(researchArea,{socialSciences,politics})>
compare(C_def(HAS-PUBLICATION), C_Q)
<(author,EMPLOYEEo<(affiliation,{research})>)
(article,PUBLICATIONo<(title,{x | substring(x) = "abortion"})>)>

Figure 20: Context Comparison: Focusing on the relevant publications

- The join condition ensures that only the articles published in the journals belonging to the areas of socialSciences and politics are considered.
- The information focusing in the object class JOURNAL is propagated to help focus information in the object class PUBLICATION.

8.3 Information Focusing: Incorporating constraints from the Query

In this section, we illustrate how information focusing occurs when a constraint specified in the query context is applied to an object class. This results in selecting only those instances from the object class which satisfy these constraints. In Figure 20, the query context C_Q is compared to C_{def}(HAS-PUBLICATION). Information is thus focused to only those publications that have the word "abortion" in their titles.

Using the rules defined in the algebra in Appendix A.4, we illustrate how the schema correspondence associated with the definition context changes as a result of the changes in the definition context. The changes in the definition context are diagrammatically illustrated in Figure 21.

1. From the example in Section 7.2.3 we have:

C_{def}(HAS-PUBLICATION)
= <(author,EMPLOYEE∘C_{ass}(EMPLOYEE,HAS-PUBLICATION))
(article,PUBLICATION∘C_{ass}(PUBLICATION,HAS-PUBLICATION))>
where C_{ass}(PUBLICATION,HAS-PUBLICATION) = <>
semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION)
= semCombine(author,semCombine(article,
semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION),
semCondition(C_{ass}(PUBLICATION,HAS-PUBLICATION),
semPro(PUBLICATION$_F$,PUBLICATION))),...)

2. The query context may be rewritten as:

C_Q = glb(<(author,X∘C_{constr}(X,ANSWER))>,Cntxt$_2$)
where C_{constr}(X,ANSWER) = <>

3. Item 1,2 and Constraint Context and Specialized Context Lifting (Rule 7.4) ⇒

semCompare(C_Q,semPro(HAS-PUBLICATION$_F$,HAS-PUBLICATION))
= comConstrain(<(author,X)>,semCompare(Cntxt$_2$,semCombine(article,...,...)),
semCompare(Cntxt$_2$,semCondition(...,...)))

4. Repeated Applications of Specialized Context Lifting and semPro Combination (Rule 7.3) and Constraint Context and Specialized Context Lifting (Rule 7.4) ⇒

semCompare(Cntxt$_2$,semCombine(article,...,...))
= comConstrain(<(article,Y$\circ C_{constr}$(Y,ANSWER))>,
semCombine(article,semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION),
semCondition(C_{ass}(PUBLICATION,HAS-PUBLICATION),
semPro(PUBLICATION$_F$,PUBLICATION))))
where C_{constr}(Y,ANSWER) = <(title,{x| substring(x)="abortion"}>

5. Constraint Incorporation (Rule 8) ⇒

semCompare(Cntxt$_2$,semCombine(article,...,...))
= semCombine(article,...,semCompare(C_{constr}(PUBLICATION,ANSWER),
semCondition(C_{ass}(PUBLICATION,HAS-PUBLICATION),
semPro(PUBLICATION$_F$,PUBLICATION))))
Specialized and Ordinary Context Lifting (Rule 7.5) ⇒
= semCombine(article,...,semCompare(
glb(C_{constr}(PUBLICATION,ANSWER),C_{ass}(PUBLICATION,HAS-PUBLICATION)),
semPro(PUBLICATION$_F$,PUBLICATION)))
= semCombine(article,...,semConstrain(<(title,{x| substring(x) = "abortion"})>,
semPro(PUBLICATION$_F$,PUBLICATION)))

6. Thus the constraint which limits the titles of the article to those that contain the word "abortion" is incorporated by the comparison of C_Q and C_{def}(HAS-PUBLICATION) and is propagated to the object class PUBLICATION.

8.4 Intra-database Correlation of Information

The representation of context and it's association with schema correspondences mapping the context to the actual storage of data in the database helps to correlate information within a database in the following ways:

- There is correlation between object classes and type definitions which are not part of the object class definition. This achieved by mapping the intensional description to the actual classes. This is discussed in detail in Section 7.2.2.
- There is correlation between an object class and other object classes which are part of the definition context of the original object class. This is discussed in detail in Section 7.2.3.
- There is correlation between two object classes due to the composition of contextual coordinates and one being a part of the definition context of the other. This is discussed in detail in Section 7.2.4.

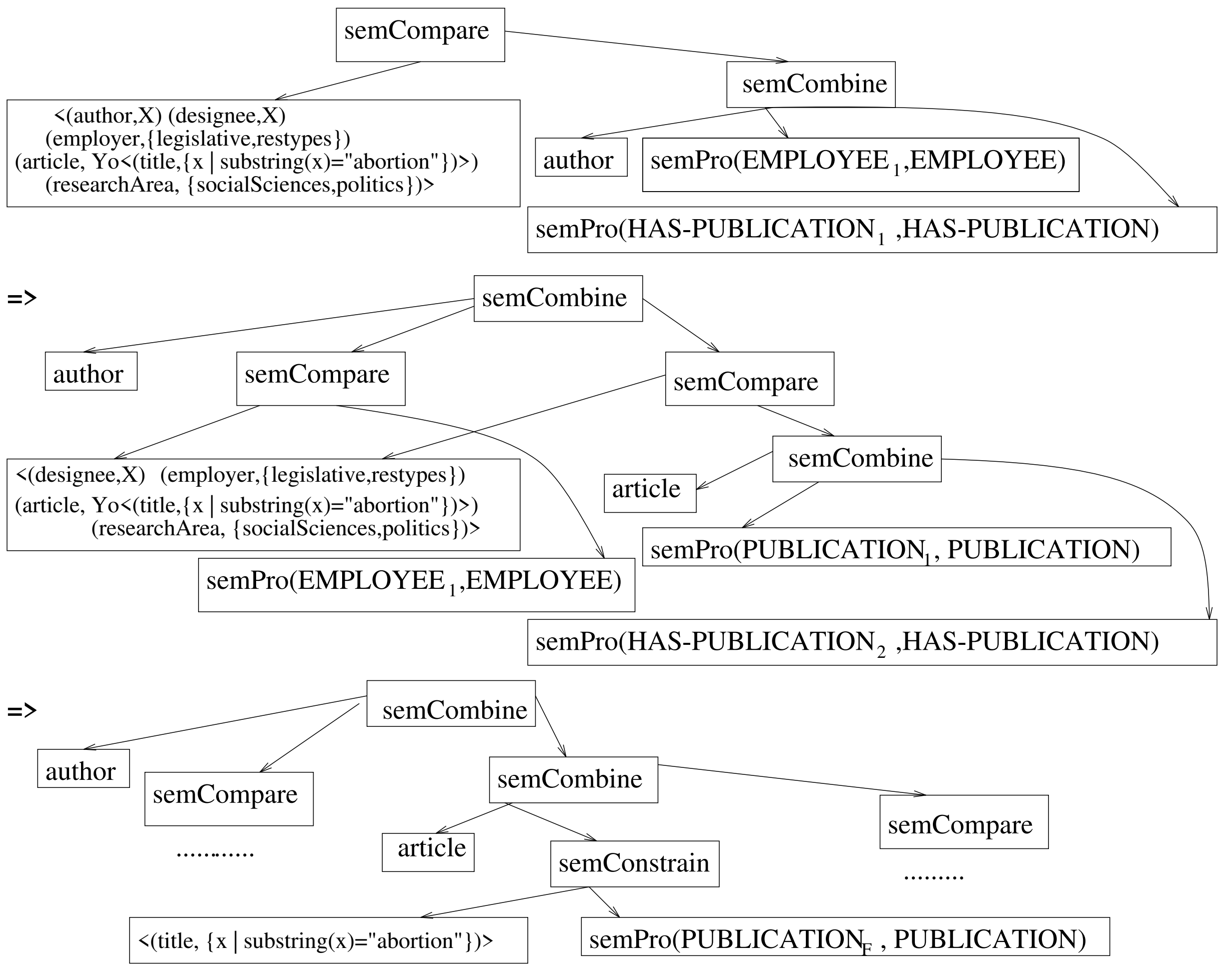

semCompare
<(author,X) (designee,X) (employer,{legislative,restypes}) (article, Yo<(title,{x | substring(x)="abortion"})>) (researchArea, {socialSciences,politics})>
semCombine
author
semPro(EMPLOYEE$_1$,EMPLOYEE)
semPro(HAS-PUBLICATION$_1$,HAS-PUBLICATION)
=>
semCombine
author
semCompare
semCompare
<(designee,X) (employer,{legislative,restypes}) (article, Yo<(title,{x | substring(x)="abortion"})>) (researchArea, {socialSciences,politics})>
semPro(EMPLOYEE$_1$,EMPLOYEE)
semCombine
article
semPro(PUBLICATION$_1$, PUBLICATION)
semPro(HAS-PUBLICATION$_2$,HAS-PUBLICATION)
=>
semCombine
author
semCompare
....
semCombine
article
semConstrain
semCompare
....
<(title, {x | substring(x)="abortion"})>
semPro(PUBLICATION$_F$, PUBLICATION)

Figure 21: Incorporating constraints from the query

8.5 Inter-database Correlation of Information

In this section, we discuss the correlation of information from different databases. This correlation is typically triggered by specifying the same variables for different contextual coordinates. These contextual coordinates might be mapped to attributes of different object classes in different databases. This is illustrated diagrammatically in Figure 22. It may be noted that the correlation can also take place in the presence of other constraints.

Example:
Consider two databases with the following object classes:
DB_1: WRITER(SS#,Name,...)
DB_2: OFFICIAL(SS#,Name,...)
ren_{WRITER}(author,SS#), ren_{WRITER}(designee,designee)
$ren_{OFFICIAL}$(author,author), $ren_{OFFICIAL}$(designee,SS#)

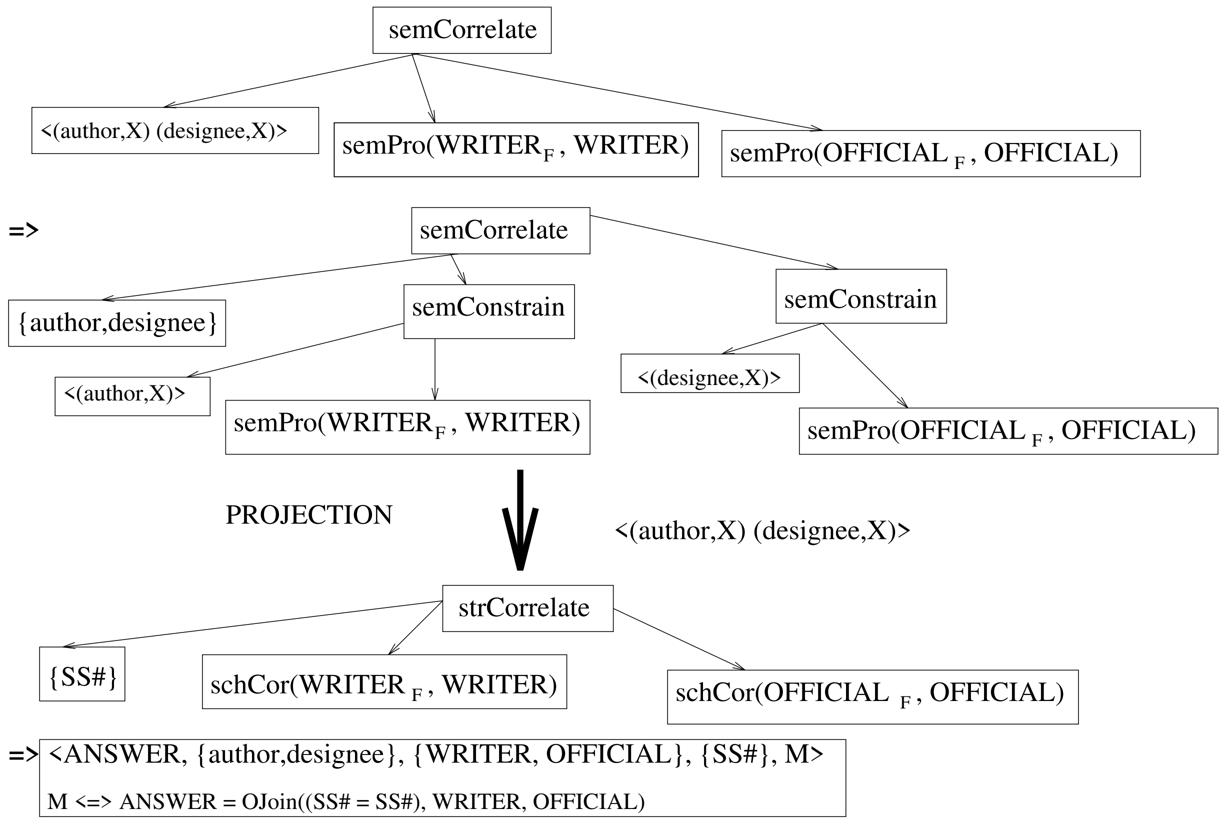
Let C_Q = <(author,X)(designee,X)>
Intuitively this models the context of a query which requires all people who are both writers and officials with a designated post. DB_1 contains information about writers and DB_2 contains information about officials. Using the rules defined in the algebra in Appendix A.4, we illustrate how the schema correspondence associated with the definition context changes as a result of the changes in the definition context. The changes in the definition context are diagrammatically illustrated in Figure 22.

1. Information Correlation (Rule 9), Specialized Constraint Lifting Application (Rule 7.2), Focusing Constraints (New Constraint, Non-existent Attribute, Rule 6.1) $\Rightarrow$

```
semCorrelate(C_Q,semPro(WRITER_F,WRITER),semPro(OFFICIAL_F,OFFICIAL))
= semCorrelate({author,designee},
                              semCompare(C_Q,semPro(WRITER_F,WRITER)),
                              semCompare(C_Q,semPro(OFFICIAL_F,OFFICIAL)))
= semCorrelate({author,designee},
                         comConstrain(<(author,X)>,semPro(WRITER_F,WRITER)),
                         comConstrain(<(designee,X)>,semPro(OFFICIAL_F,OFFICIAL)))
```

2. Information Correlation and Projection (Rule 9) $\Rightarrow$

```
Π_{C_Q}(semCorrelate({author,designee},
                         comConstrain(<(author,X)>,semPro(WRITER_F,WRITER)),
                         comConstrain(<(designee,X)>,semPro(OFFICIAL_F,OFFICIAL))))
= strCorrelate({ren_WRITER(author,SS#),ren_OFFICIAL(designee,SS#)},
                  schCor(OFFICIAL_F,OFFICIAL),schCor(WRITER_F,WRITER))
= <ANSWER,{author,designee},{WRITER,OFFICIAL},{SS#},M>
M ≡ ANSWER=OJoin((SS#=SS#),WRITER,OFFICIAL)
```



semCorrelate
<(author,X) (designee,X)>
semPro(WRITER_F, WRITER)
semPro(OFFICIAL_F, OFFICIAL)
=>
semCorrelate
{author,designee}
semConstrain
<(author,X)>
semPro(WRITER_F, WRITER)
semConstrain
<(designee,X)>
semPro(OFFICIAL_F, OFFICIAL)
PROJECTION
<(author,X) (designee,X)>
strCorrelate
{SS#}
schCor(WRITER_F, WRITER)
schCor(OFFICIAL_F, OFFICIAL)
=> <ANSWER, {author,designee}, {WRITER, OFFICIAL}, {SS#}, M>
M <=> ANSWER = OJoin((SS# = SS#), WRITER, OFFICIAL)

Figure 22: Inter-database correlation of information

9 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 have taken cues from various fields of research such as AI, knowledge representation, cognitive psychology and linguistics to make a case for the explicit identification and representation of context in a multidatabase environment. We also discuss the inadequacy of structural similarity and how semantics cannot be captured by purely mathematical formalisms. This lead us to 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 object classes.

Using the framework of semantic proximity and schema correspondences, we demonstrate the reconciliation of the dual schematic *vs* semantic perspective. This is done by associating the schema correspondence(s) with the context of the semantic proximity among the object classes. This association enables us to determine measures of semantic similarity viz. *equivalence, relationship, relevance, resemblance and incompatibility* and develop a semantic taxonomy. We also enumerate the various schematic heterogeneities and the possible the semantic similarities between them.

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

Besides helping to reconcile the semantic and the structural perspectives, it also enables us to perform query processing. It enables *information focusing* as any changes to the context affect the schema correspondences and help retrieve only the data relevant to the query. It enables *information correlation* as one can specify constraints relating different object classes in the context. The computation of the resulting schema correspondences enables the correlation of the appropriate instances of the object classes. The association of the schema correspondences with the context also enables us to capture associations and information which may not be captured in the database.

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 object classes in a database, issues of language and vocabulary become important. In designing the definition context of an object class, 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.

[BN75] D. Bobrow and D. Norman. Some principles of Memory Schemata. In *Representation and Understanding*. New York : Academic Press, 1975.

[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.

[BW85] D. Bobrow and T. Winograd. An overview of KRL, a Knowledge Representation Language. In *Readings in Knowledge Representation*. Morgan Kaufmann, 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.

[HM85] D. Heimbigner and D. McLeod. A federated architecture for Information Systems. *ACM Transactions on Office Information Systems*, 3,3, 1985.

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

[Ken91] W. Kent. The breakdown of the Information Model in Multidatabase Systems. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[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.

[KS] V. Kashyap and A. Sheth. Scalable Semantics-based Information Brokering: Problems in Language and Vocabulary. In preparation ....

[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.

[KS94] V. Kashyap and A. Sheth. Semantics-based Information Brokering. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM)*, November 1994.

[KS95] 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.

[MC91] G. A. Miller and W. G. Charles. Contextual Correlates of Semantic Similarity. *Languauge and Cognitive processes*, 1991.

[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.

[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.

[SGN93] A. Sheth, S. Gala, and S. Navathe. On automatic Reasoning for Schema Integration. *International Journal on Intelligent and Cooperative Information Systems*, 2(1), March 1993.

[She91a] A. Sheth. Federated Database Systems for managing Distributed, Heterogeneous, and Autonomous Databases. *Tutorial Notes - the 17th VLDB Conference*, September 1991.

[She91b] 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.

[Tho89] J. P. Thompson. *Data with Semantics : Data Models and Data Management.* Van Nostrand Reinhold - New York, 1989.

[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. Ontology Algebra. In *FGCS Workshop on Heterogeneous Cooperative Knowledge-Bases*, December 1994.

[Woo85] J. Wood. What's in a link ? In *Readings in Knowledge Representation.* Morgan Kaufmann, 1985.

[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 Relevant terminology and operations

ren$_O$(C,A) This is the *rename* operator which stores the association between a contextual coordinate C from the ontology and an attribute A of an object class O if there exists one.

semConstrain(<(C$_i$,V$_i$)>,semPro(O$_1$,O$_2$)) This operator poses the constraint represented by a contextual coordinate and it's value on the semPro descriptor, thus modifying it.

strConstrain(ren$_{O_2}$(C$_j$,A$_j$),S$_j$,schCor(O$_1$,O$_2$)) The structural counterpart of semConstrain. It actually maps the contextual coordinates to the attributes and recomputes the mappings.

semCondition(Cntxt,semPro(O$_1$,O$_2$)) This operator modifies the semantic proximity descriptor by *lifting* [Guh91] it into a context different from which it is defined in. The result of this operator is a modified semPro descriptor between O$_1$ and O$_2$.

semCombine(C$_i$,semPro(O$_1$,O$_2$),semPro(O$_3$,O$_i$)) This operator is used to combine information when the definition context of an object class depends on other object classes. The result of this operator is also a modified semPro descriptor between O$_1$ and O$_2$.

strCombine(strCombine({ren$_{O_2}$(C$_i$,A'$_i$),ren$_{O_i}$(C$_i$,A'$_i$)},schCor(O$_1$,O$_2$),semPro(O$_3$,O$_j$)) The structural counterpart of semCombine. Actually maps the contextual coordinate to the attributes of the object classes and correlates the instances bases on this information.

comConstrain<(C$_i$,V$_i$)>,semPro(O$_1$,O$_2$)) This is similar to the semConstrain operation but for one difference. It has no affect on the semPro descriptor when C$_i$ is not present in the context in which semPro(O$_1$,O$_2$) is defined and is not associated with any attribute of the object class.

semCompare(Cntxt,semPro(O$_1$,O$_2$)) This is similar to semCondition except for the fact that the contextual coordinates which are not present in the context in which semPro(O$_1$,O$_2$) are defined or are not associated with any attribute of the object class will have no effect on the semPro descriptor.

semCorrelate(Cntxt,semPro(O$_1$,O$_2$),semPro(O$_3$,O$_4$)) Correlates information corresponding to different object classes related through some constraint in the correlation context.

strCorrelate($\{$ren$_{O_1}$(C$_1$,A$_1$),ren$_{O_2}$(C$_2$,A$_2$)$\}$,schCor(O$_1$,O$_2$),schCor(O$_3$,O$_4$)) The structural counterpart of semCorrelate. Actually maps the contextual coordinates to the attributes and correlates the instances of the classes based on that information.

A.2 Object Algebraic Operations

In this section we list a limited set of operations to manipulate object classes in a database and specify their semantics. These operations are a modification of those identified in [SZ90]. Object classes are considered as collections of objects which are homogeneous and have the same type as the abstract data type associated with the class. The properties exported by the interface of the abstract data type are the attributes of the object class. We are primarily concerned with operations that retrieve data, though there are some operations which create a new object.

OSelect(p,O) This is the select operation which satisfies a set of database objects satisfying a selection predicate, p.

$$\text{OSelect(p,O)} = \{o \mid o \in O \wedge p(o)\}$$

makeObjectClass(C,S) Given a contextual coordinate C and a set S (which maybe either a set of values from the ontology, object class of type domain) defines a new object class with instances having attribute C and a value from the set S as it's value.

$$\text{makeObjectClass(C,S)} = \{o \mid o.C{=}s \wedge s \in S\}$$

OProduct(O$_1$,O$_2$) Given two object classes O$_1$ and O$_2$, a new object class 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.

$$\text{OProduct[}(O_1,O_2) = \{o \mid o.A_i{=}o_1.A_i \wedge A_i \in \text{attr}(O_1) \wedge o_1 \in O_1 \wedge o.A_j{=}o_2.A_j \wedge A_j \in \text{attr}(O_2) \wedge o_2 \in O_2\}$$

OJoin(p,O$_1$,O$_2$) This can be thought of as a special case of the operator OProduct, except that the instances should satisfy the predicate p, specified.

$$\text{OJoin}(p,O_1,O_2) = \{o \mid o \in \text{OProduct}(O_1,O_2) \wedge p(o)\}$$

OUnion(O$_1$,O$_2$) This is the standard set union operation modified for object classes.

$$\text{OUnion}(O_1,O_2) = \{o \mid o \in O_1 \vee o \in O_2\}$$

OIntersect(O$_1$,O$_2$) This is the standard set intersection operation modified for object classes.

$$\text{OIntersect}(O_1,O_2) = \{o \mid o \in O_1 \wedge o \in O_2\}$$

A.3 Algebra for the Projection Operation

semPro(O_1,O_2) = <Cntxt,M,(dom(O_1),dom(O_2)),_>

Π_{Cntxt}(semPro(O_1,O_2)) = schCor(O_1,O_2) = <O_1,attr(O_1),O_2,attr(O_2),M>

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

```
/* When semPro is defined with an empty context, the object is returned unchanged */
```

$$\boxed{\begin{array}{l}\text{schCor}(O_1,O_2) = <O_1,\phi,O_2,\phi,M> \\ M \equiv O_1{=}O_2\end{array}}$$

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

```
/* When the values of the contextual coordinates are sets of symbols from the */
/* Ontology and each contextual coordinate is associated with an attribute    */
```

$$\boxed{\begin{array}{l}\text{schCor}(O_1,O_2) = <O_1,\{C_i|C_i \in \text{Cntxt}\},O_2,\{\text{ren}_{O_2}(C_i,A_i)|C_i \in \text{Cntxt}\},M> \\ M \equiv O_1{=}\text{OSelect}(p,O_2), \text{ where } p \equiv A_1 \in S_1 \wedge ... \wedge A_k \in S_k\end{array}}$$

Rule 3: *Constraint Application*, when Cntxt = glb(<(C_j,S_j)>,Cntxt_1)

```
/* Each of the contextual coordinates defines a constraint which is applied   */
/* applied till an empty context is obtained       */
```

$$\boxed{\begin{array}{l}\text{semPro}(O_1,O_2) = \text{semConstrain}(<(C_j,S_j)>, \text{semPro}(O_3,O_2)) \\ \text{where semPro}(O_3,O_2) = <\text{Cntxt}_1,M,(\text{dom}(O_3),\text{dom}(O_2)),\_>\end{array}}$$

Constraint Application Projection

```
/* defines the appropriate mapping of the constraint in terms of the database */
/* objects in a recursive manner       */
```

$$\boxed{\begin{array}{l}\Pi_{Cntxt}(\text{semConstrain}(<(C_j,S_j)>, \text{semPro}(O_3,O_2)) \\ = \text{strConstrain}(\text{ren}_{O_2}(C_j,A_j),S_j,\Pi_{Cntxt_1}(\text{semPro}(O_3,O_2)))\end{array}}$$

where Π_{Cntxt_1}(semPro(O_3,O_2)) is given by:

$$\boxed{\begin{array}{l}\text{schCor}(O_3,O_2) \\ = <O_3,\{C_i|C_i \in \text{Cntxt}_1\},O_2,\{\text{ren}_{O_2}(C_i,A_i)|C_i \in \text{Cntxt}_1\},M_3 > \\ M_3 \equiv O_3{=}\text{OSelect}(p,O_2)\end{array}}$$

Thus, Π_{Cntxt}(semPro(O_1,O_2)) = strConstrain(ren_{O_2}(C_j,A_j),S_j,schCor(O_3,O_2))

Rule 3.1: *New Constraint, Existent Attribute*, i.e. $C_j \notin \text{Cntxt}_1$, ren_{O_2}(C_j,A_j) exists.

```
/* additional coordinate is not present in the context and there exists an    */
/* association between the contextual coordinate and an attribute of object   */
```

$$\boxed{\begin{array}{l}\text{strConstrain}(\text{ren}_{O_2}(C_j,A_j),S_j,\text{schCor}(O_3,O_2)) \\ = <O_1,\{C_j\}\cup\{C_i|C_i \in \text{Cntxt}_1\},O_2,\{\text{ren}_{O_2}(C_j,A_j)\}\cup\{\text{ren}_{O_2}(C_i,A_i)|C_i \in \text{Cntxt}_1\},M> \\ M \equiv O_1{=}\text{OSelect}((A_j \in S_j),O_3){=}\text{OSelect}(p\wedge(A_j \in S_j),O_2)\end{array}}$$

Rule 3.2: *Modified Constraint, Existent Attribute*, i.e. $C_j \in Cntxt_1$, $ren_{O_2}(C_j,A_j)$ exists

```
/* additional coordinate is present in the context which results in the      */
/* modification of the constraint after which it becomes similar to Rule 3.1  */
```

$Cntxt_1=glb(<(C_j,S'_j)>,Cntxt_2)$
$\Rightarrow glb(<(C_j,S_j)>,Cntxt_1)=glb(<(C_j,S_j \cap S'_j)>,Cntxt_2)$ where $C_j \notin Cntxt_2$
$\Rightarrow$ Rule 3.1 can now be applied

Rule 3.3: *New Constraint, Non-existent attribute*, i.e. $C_j \notin Cntxt_1$, $ren_{O_2}(C_j,A_j)$ does not exist

```
/* domain augmentation of the object class takes place as the contextual      */
/* coordinate is not associated with any attribute of the object class        */
```

$strConstrain(ren_{O_2}(C_j,C_j),S_j,schCor(O_3,O_2))$
$= <O_1,\{C_j\}\cup\{C_i|C_i \in Cntxt_1\},O_2,\{ren_{O_2}(C_j,C_j)\}\cup\{ren_{O_2}(C_i,A_i)|C_i \in Cntxt_1\},M>$
$M \equiv O_1=OProduct(makeObjectClass(C_j,S_j),O_3)$
$=OProduct(makeObjectClass(C_j,S_j),OSelect(p,O_2))$

Rule 3.4: *Modified Constraint, Non-existent Attribute*, i.e. $C_j \in Cntxt_1$, $ren_{O_2}(C_j,A_j)$ does not exist

```
/* additional contextual coordinate is present in the context which leads to  */
/* constraint modification after which it becomes similar to Rule 3.3         */
```

$Cntxt_1=glb(<(C_j,S'_j)>,Cntxt_2)$
$\Rightarrow glb(<(C_j,S_j)>,Cntxt_1)=glb(<(C_j,S_j \cap S'_j)>,Cntxt_2)$ where $C_j \notin Cntxt_2$
$\Rightarrow$ Rule 3.3 can now be applied

Rule 4: *Context Lifting and Projection*, i.e. $semCondition(Cntxt_1,semPro(O_1,O_2))$

```
/* when the semPro descriptor is modified by evaluating it in a context which */
/* is different from the context in which it is defined                       */
```

$\Pi_{Cntxt_1}(semCondition(Cntxt_1,semPro(O_1,O_2))) = \Pi_{glb(Cntxt_1,Cntxt)}(semPro(O_1,O_2))$

Rule 4.1: *Empty Context Lifting and Projection*, i.e. $Cntxt_1 = <>$

```
/* when a semPro descriptor is lifted to an empty context there is no change  */
/* in the semPro and the associated schCor descriptors      */
```

$semCondition(Cntxt_1,semPro(O_1,O_2)) = semPro(O_1,O_2)$
$\Pi_{glb(Cntxt_1,Cntxt)}(semPro(O_1,O_2)) = schCor(O_1,O_2)$

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

```
/* each contextual coordinate represents a constraint which is recursively    */
/* used to modify the semPro descriptor till the context becomes empty        */
```

$semCondition(Cntxt_1,semPro(O_1,O_2))$
$= semConstrain(<(C_j,S_j)>,semCondition(Cntxt_2,semPro(O_1,O_2)))$

Constraint Lifting Application Projection

$\Pi_{Cntxt_1}(semConstrain(<(C_j,S_j)>,semCondition(Cntxt_2,semPro(O_1,O_2))))$
$= strConstrain(ren_{O_2}(C_j,A_j),S_j,\Pi_{Cntxt_2}(semCondition(Cntxt_2,semPro(O_1,O_2))))$

Rule 4.3: *Context Lifting and semPro Combination*, i.e.
semCondition(Cntxt$_1$,semCombine(C$_i$,semPro(O$_3$,O$_2$),semPro(O$_4$,O$_j$)))

```
/* When a semPro which is a combination is lifted to a different context,    */
/* each of it's component semPro descriptors are also lifted to that context  */
/* The combination of semPro descriptors is defined in Rule 5       */
```

$$\boxed{\begin{array}{l} = \text{semCombine}(C_i,\text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_3,O_2)), \\ \qquad\qquad \text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_4,O_j))) \end{array}}$$

Context Lifting and semPro Combination Projection

$$\boxed{\begin{array}{l} \Pi_{Cntxt_1}(\text{semCombine}(C_i,\text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_3,O_2)), \\ \qquad\qquad \text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_4,O_j)))) \\ = \text{strCombine}(\{\text{ren}_{O_2}(C_i,A'_i),\text{ren}_{O_j}(C_i,A'_i)\},\Pi_{Cntxt_1}(\text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_3,O_2))), \\ \qquad\qquad \Pi_{Cntxt_1}(\text{semCondition}(\text{Cntxt}_1,\text{semPro}(O_4,O_j)))) \end{array}}$$

Rule 5: *semPro Combination*, i.e. Cntxt = glb(<(C$_j$,O$_j\circ$C$_{ass}$(O$_j$,O$_2$))>,Cntxt$_1$)

```
/* when the value of a contextual coordinate is an object class associated    */
/* an association context. This results in a combination of two semPros       */
```

$$\boxed{\begin{array}{l} \text{semPro}(O_1,O_2) = \text{semConstrain}(<(C_j,O_j\circ C_{ass}(O_j,O_2))>,\text{semPro}(O_3,O_2)) \\ = \text{semCombine}(C_j,\text{semPro}(O_3,O_2),\text{semCondition}(C_{ass}(O_j,O_2),\text{semPro}(O_{jF},O_j))) \end{array}}$$

semPro Combination Projection

$$\boxed{\begin{array}{l} \Pi_{Cntxt}(\text{semCombine}(C_j,\text{semPro}(O_3,O_2),\text{semCondition}(C_{ass}(O_j,O_2),\text{semPro}(O_{jF},O_j)))) \\ = \text{strCombine}(\{\text{ren}_{O_j}(C_j,A_j),\text{ren}_{O_2}(C_j,A'_j)\},\Pi_{Cntxt_1}(\text{semPro}(O_3,O_2)), \\ \qquad\qquad \Pi_{C_{ass}(O_j,O_2)}(\text{semCondition}(C_{ass}(O_j,O_2),\text{semPro}(O_{jF},O_j)))) \end{array}}$$

where Π_{Cntxt_1}(semPro(O$_3$,O$_2$)) may be given as:

$$\boxed{\begin{array}{l} \text{schCor}(O_3,O_2)) \\ = <O_3,\{C_i|C_i \in\text{Cntxt}_1\},O_2,\{\text{ren}_{O_2}(C_i,A_i)|C_i \in\text{Cntxt}_1\},M_3 > \\ M_3 \equiv O_3=\text{OSelect}(p,O_2) \end{array}}$$

Constraint Lifting Application Projection Rule ⇒

$$\boxed{\begin{array}{l} \Pi_{C_{ass}(O_j,O_2)}(\text{semCondition}(C_{ass}(O_j,O_2),\text{semPro}(O_{jF},O_j))) \\ = \text{schCor}(O_4,O_j)) \\ = <O_4,\{C_i|C_i \in C_{ass}(O_j,O_2)\vee C_i \in C_{def}(O_j)\},O_j, \\ \qquad\qquad \{\text{ren}_{O_j}(C_i,A_i)|C_i \in C_{ass}(O_j,O_2)\vee C_i \in C_{def}(O_j)\},M_4 > \\ M_4 \equiv O_4=\text{OSelect}(p',O_j) \end{array}}$$

Thus we have:

$$\boxed{\begin{array}{l} \Pi_{Cntxt}(\text{semCombine}(C_j,\text{semPro}(O_3,O_2),\text{semCondition}(C_{ass}(O_j,O_2),\text{semPro}(O_{jF},O_j)))) \\ = \text{strCombine}(\{\text{ren}_{O_j}(C_j,A_j),\text{ren}_{O_2}(C_j,A'_j)\},\text{schCor}(O_3,O_2),\text{schCor}(O_4,O_j)) \end{array}}$$

Rule 5.1: *New Constraint and Existent Attributes*, i.e. $C_j \notin Cntxt_1$, $ren_{O_j}(C_j,A_j)$ and $ren_{O_2}(C_j,A'_j)$ exist

```
/* additional coordinate is mapped into attributes for both the objects       */
/* and is not present in the original context       */
```

$$\begin{array}{l} strCombine(\{ren_{O_j}(C_j,A_j),ren_{O_2}(C_j,A'_j)\},schCor(O_3,O_2),schCor(O_4,O_j)) \\ = <O_1,\{C_j\}\cup\{C_i|C_i \in Cntxt_1\},\{O_2,O_j\} \\ \qquad \{ren_{O_j}(C_j,A_j),ren_{O_2}(C_j,A'_j)\}\cup\{ren_{O_2}(C_i,A_i)|C_i \in Cntxt_1\},M> \\ M \equiv O_1=OJoin(g(A_j,A'_j),O_3,O_4) \\ \qquad =OJoin(g(A_j,A'_j),OSelect(p,O_2),OSelect(p',O_j)) \end{array}$$

Rule 5.2: *Modified Constraint and Existent Attributes*, i.e. $C_j \in Cntxt_1$, $ren_{O_j}(C_j,A_j)$ and $ren_{O_2}(C_j,A'_j)$ exist

```
/* additional coordinate is already present in the context leads to          */
/* constraint modification after which Rule 5.1 can be applied       */
```

$$\begin{array}{l} Cntxt_1=glb(<(C_j,V_j)>,Cntxt_2) \\ \Rightarrow glb(<(C_j,O_j \circ C_{ass}(O_j,O_2))>,Cntxt_1)=glb(<(C_j,glb(V_j,O_j \circ C_{ass}(O_j,O_2)))>,Cntxt_2) \\ \quad \text{where } C_j \notin Cntxt_2 \\ \Rightarrow \text{Rule 5.1 can be applied} \end{array}$$

Rule 5.3 *Contextual Coordinate Composition*, i.e. C_j = compose($C_{j,1}$,$C_{j,2}$)

```
/* when a contextual coordinate is the composition of two or more contextual  */
/* coordinates. Each may or may not be mapped into attributes       */
```

The composition of attributes is as follows:

$$\begin{array}{l} ren_O(C_j,X) = ren_O(compose(C_{j,1},C_{j,2}),compose(X_1,X_2)) \\ = compose(ren_O(C_{j,1},X_1),ren_O(C_{j,2},X_2)) \\ \text{Let } ren_{O_2}(C_j,A'_j) = compose(ren_{O_2}(C_{j,1},A'_{j,1}),ren_{O_2}(C_{j,2},A'_{j,2})) \\ \text{Let } ren_{O_j}(C_j,A_j) = compose(ren_{O_j}(C_{j,1},A_{j,1}),ren_{O_j}(C_{j,2},A_{j,2})) \end{array}$$

Contextual Coordinate Composition Projection

$$\begin{array}{l} strCombine(\{ren_{O_j}(C_j,A_j),ren_{O_2}(C_j,A'_j)\},schCor(O_3,O_2),schCor(O_4,O_j)) \\ = <O_1,\{C_j\}\cup\{C_i|C_i \in Cntxt_1\},\{O_2,O_j\} \\ \qquad \{ren_{O_j}(C_j,A_j),ren_{O_2}(C_j,A'_j)\}\cup\{ren_{O_2}(C_i,A_i)|C_i \in Cntxt_1\},M> \\ M \equiv O_1=OJoin(g(<A_{j,1},A_{j,2}>,<A'_{j,1},A'_{j,2}>),O_3,O_4) \\ \qquad =OJoin(g(<A_{j,1},A_{j,2}>,<A'_{j,1},A'_{j,2}>),OSelect(p,O_2),OSelect(p',O_j)) \end{array}$$

A.4 An Algebra for focusing and correlation

In this section we given algebra to express the changes in the schema correspondences when the contexts they are associated with change. The changes in contexts are represented by operations used to manipulate contexts.

Rule 6: *Focusing Constraints*, i.e. Cntxt = compare(<(C_j,S_j)>,$Cntxt_1$)

```
/* similar to the constraint application operation defined in section A.3 but */
/* for a few differences enumerated in the following rules       */
```

comConstrain(<(C_j,S_j)>,semPro(O_1,O_2))
where semPro(O_1,O_2) = <$Cntxt_1$,M,(dom(O_1),dom(O_2)),_>

Focusing Constraints Projection

Π_{Cntxt}(comConstrain(<(C_j,S_j)>,semPro(O_1,O_2))
= strConstrain(ren_{O_2}(C_j,A_j),S_j,Π_{Cntxt_1}(semPro(O_1,O_2)))

where Π_{Cntxt_1}(semPro(O_1,O_2)) may be given by:

schCor(O_1,O_2)
= <O_1,{C_i|C_i ∈$Cntxt_1$},O_2,{ren_{O_2}(C_i,A_i)|C_i ∈$Cntxt_1$},M_3 >
M_3 ≡ O_1=OSelect(p,O_2)

Rule 6.1: *New Constraint, Non-existent Attribute*, i.e. C_j ∉$Cntxt_1$,ren_{O_2}(C_j,A_j) does not exist

```
/* If the additional coordinate is not associated with any attribute of the   */
/* object class, it is ignored and there is no change in semPro and schCor    */
```

comConstrain(<(C_j,S_j)>,semPro(O_3,O_2)) = semPro(O_3,O_2)
Π_{Cntxt}(comConstrain(<(C_j,S_j)>,semPro(O_1,O_2))) = schCor(O_2,O_2)

Rule 6.2: *Modified Constraint, Existent Attribute*, i.e. C_j ∈$Cntxt_1$, ren_{O_2}(C_j,A_j) exists

```
/* additional coordinate is present in the context leads to the modification  */
/* of the constraint and since there is an attribute associated with the      */
/* contextual coordinate Rule 3.1 can be applied       */
```

$Cntxt_1$=glb(<(C_j,S'$_j$)>,$Cntxt_2$)
⇒ compare(<(C_j,S_j)>,$Cntxt_1$)=glb(<(C_j,S_j∩S'$_j$)>,$Cntxt_2$) where C_j ∉$Cntxt_2$
Constrain Application Rule, Existent Attribute case ⇒
comConstrain(<(C_j,S_j)>,semPro(O_1,O_2)) = semConstrain(<(C_j,S_j)>,semPro(O_1,O_2))

Rule 6.3: *Modified Constraint, Non-existent Attribute*, i.e. C_j ∈$Cntxt_1$, ren_{O_2}(C_j,A_j) does not exist

```
/* additional coordinate is present in the context leads to modification of   */
/* constraint        */
```

$Cntxt_1$=glb(<(C_j,S'$_j$)>,$Cntxt_2$)
⇒ compare(<(C_j,S_j)>,$Cntxt_1$)=glb(<(C_j,S_j∩S'$_j$)>,$Cntxt_2$) where C_j ∉$Cntxt_2$
Constraint Application Rule, New Constraint, Non-existent Attribute case⇒
comConstrain(<(C_j,S_j)>,semPro(O_1,O_2)) = semConstrain(<(C_j,S_j)>,semPro(O_1,O_2))

Rule 6.4: *New Constraint, Existent attribute*, i.e. $C_j \notin Cntxt_1$,$ren_{O_2}(C_j,A_j)$ exists

```
/* This a new constraint and there is an attribute associated with the       */
/* contextual coordinate which makes it exactly similar to Rule 3.1      */
```

Constraint Application Rule, New Constraint, Existent Attribute $\Rightarrow$
comConstrain(<(C_j,S_j)>,semPro(O_1,O_2)) = semConstrain(<(C_j,S_j)>,semPro(O_1,O_2))

Rule 7: *Specialized Context Lifting and Projection*, i.e. semCompare($Cntxt_1$,semPro(O_1,O_2))

```
/* similar to the context lifting operation defined in A.3 but with a few     */
/* differences shown below      */
```

Π_{Cntxt_1}(semCompare($Cntxt_1$,semPro(O_1,O_2))) = $\Pi_{compare(Cntxt_1,Cntxt)}$(semPro($O_1$,$O_2$))

Rule 7.1: *Empty Specialized Context Lifting and Projection*, i.e. $Cntxt_1$ = <>

```
/* the empty context case in which the semPro and schCor remain unchanged */
```

semCompare($Cntxt_1$,semPro(O_1,O_2)) = semPro(O_1,O_2)
$\Pi_{compare(Cntxt_1,Cntxt)}$(semPro($O_1$,$O_2$)) = schCor($O_1$,$O_2$)

Rule 7.2: *Specialized Constraint Lifting Application and Projection*, i.e. $Cntxt_1$ = glb(<(C_j,S_j)>,$Cntxt_2$)

```
/* each contextual coordinate represents a constraint which is recursively    */
/* used to change semPro till the context becomes empty, the comConstrain     */
/* operation is used instead of the semConstrain operation       */
```

semCompare($Cntxt_1$,semPro(O_1,O_2))
= comConstrain(<(C_j,S_j)>,semCompare($Cntxt_2$,semPro(O_1,O_2)))

Rule 7.3: *Specialized Context Lifting and semPro Combination*, i.e. $C_i \notin Cntxt_1$ and semCompare($Cntxt_1$,semCombine(C_i,semPro(O_3,O_2),semPro(O_4,O_j)))

```
/* when a semPro which is a combination is lifted to a different context and  */
/* the additional coordinate is not present in the lifting context, the       */
/* component semPro descriptors are also lifted to that context.      */
```

= semCombine(C_i,semCompare($Cntxt_1$,semPro(O_3,O_2)),
semCompare($Cntxt_1$,semPro(O_4,O_j)))

Specialized Context Lifting and semPro Combination Projection

Π_{Cntxt_1}(semCombine(C_i,semCompare($Cntxt_1$,semPro(O_3,O_2)),
semCompare($Cntxt_1$,semPro(O_4,O_j))))
= strCombine({ren_{O_2}(C_i,A'$_i$),ren_{O_j}(C_i,A'$_i$)},Π_{Cntxt_1}(semCompare($Cntxt_1$,semPro(O_3,O_2))),
Π_{Cntxt_1}(semCompare($Cntxt_1$,semPro(O_4,O_j))))

Rule 7.4: *Constraint Context and Specialized Context Lifting*, i.e.
$\mathrm{Cntxt}_1 = \mathrm{glb}(<(C_i, X \circ C_{constr}(X,\mathrm{ANSWER})>, \mathrm{Cntxt}_2)$

```
/* Corresponds to the case where the value of a contextual coordinate consist */
/* of a variable associated with a constraint context      */
```

$$\boxed{\begin{array}{l} \mathrm{semCompare}(\mathrm{Cntxt}_1,\mathrm{semCombine}(C_i,\mathrm{semPro}(O_3,O_2),\mathrm{semPro}(O_4,O_j))) \\ = \mathrm{comConstrain}(<(C_i,X\circ C_{constr}(X,\mathrm{ANSWER})>, \\ \qquad\qquad (\mathrm{semCombine}(C_i,\mathrm{semCompare}(\mathrm{Cntxt}_2,\mathrm{semPro}(O_3,O_2)), \\ \qquad\qquad\qquad \mathrm{semCompare}(\mathrm{Cntxt}_2,\mathrm{semPro}(O_4,O_j))) \end{array}}$$

Rule 7.5: *Specialized and Ordinary Context Lifting*

```
/* models the interaction between the two types of context lifting operations */
```

$$\boxed{\begin{array}{l} \mathrm{semCompare}(\mathrm{Cntxt}_1,\mathrm{semCondition}(\mathrm{Cntxt}_2,\mathrm{semPro}(O_1,O_2))) \\ = \mathrm{semCompare}(\mathrm{glb}(\mathrm{Cntxt}_1,\mathrm{Cntxt}_2),\mathrm{semPro}(O_1,O_2)) \\ \Rightarrow \text{Rule 7.2 can be applied} \end{array}}$$

Rule 8: *Constraint Incorporation*

```
/* Arises when the value of the additional contextual coordinate is variable  */
/* associated with a constraint context is used to modify a semPro which is a */
/* combination of semPros       */
```

$$\boxed{\begin{array}{l} \mathrm{comConstrain}(<(C_j,X\circ C_{constr}(X,\mathrm{ANSWER}))>,\mathrm{semPro}(O_1,O_2)) \text{ where } \mathrm{semPro}(O_1,O_2) \\ = \mathrm{semCombine}(C_j,\mathrm{semPro}(O_3,O_2),\mathrm{semCondition}(C_{ass}(O_j,O_2),\mathrm{semPro}(O_{jF},O_j))) \end{array}}$$

The resulting modification of the semPro is:

$$\boxed{\begin{array}{l} \mathrm{comConstrain}(<(C_j,X\circ C_{constr}(X,\mathrm{ANSWER}))>,\mathrm{semPro}(O_1,O_2)) \\ = \mathrm{semCombine}(C_j,\mathrm{semCompare}(C_{constr}(X,\mathrm{ANSWER}),\mathrm{semPro}(O_3,O_2)), \\ \qquad \mathrm{semCompare}(C_{constr}(X,\mathrm{ANSWER}),\mathrm{semCondition}(C_{ass}(O_j,O_2),\mathrm{semPro}(O_{jF},O_j))) \\ \textsf{Assumption: There is no overlap between the context in which semPro}(O_3,O_2) \textsf{ is defined and} \\ C_{constr}(\mathrm{X,ANSWER}) \\ = \mathrm{semCombine}(C_j,\mathrm{semPro}(O_3,O_2), \\ \qquad \mathrm{semCompare}(C_{constr}(O_j,\mathrm{ANSWER}),\mathrm{semCondition}(C_{ass}(O_j,O_2),\mathrm{semPro}(O_{jF},O_j))) \end{array}}$$

Rule 9: *Information Correlation*, i.e. $\mathrm{Cntxt} = \mathrm{glb}(\mathrm{Cntxt}_1,\mathrm{Cntxt}_2)$
$\mathrm{Cnxt}_1 = <(C_1, X\circ C_{constr1}(X,\mathrm{ANSWER}))>$
$\mathrm{Cnxt}_2 = <(C_2, X\circ C_{constr2}(X,\mathrm{ANSWER}))>$
$\mathrm{ren}_{O_1}(C_1,A_1)$, $\mathrm{ren}_{O_2}(C_1,C_1)$
$\mathrm{ren}_{O_1}(C_2,C_2)$, $\mathrm{ren}_{O_2}(C_2,A_2)$

```
/* models the correlation of information as a result of satisfying the        */
/* constraints on variables in a context (equality constraint in this case)    */
```

$$\boxed{\begin{array}{l} \mathrm{semCorrelate}(\mathrm{Cntxt},\mathrm{semPro}(O_{1F},O_1),\mathrm{semPro}(O_{2F},O_2)) \\ = \mathrm{semCorrelate}(\{C_1,C_2\},\mathrm{semCompare}(\mathrm{Cntxt}_1,\mathrm{semPro}(O_{1F},O_1)), \\ \qquad\qquad \mathrm{semCompare}(\mathrm{Cntxt}_2,\mathrm{semPro}(O_{2F},O_2))) \end{array}}$$

Information Correlation Projection

Π_{Cntxt}(semCorrelate(Cntxt,semPro(O_{1F},O_1),semPro(O_{2F},O_2)))
= strCorrelate({ren_{O_1}(C_1,A_1),ren_{O_2}(C_2,A_2)},Π_{Cntxt_1}(semCompare(Cntxt_1,semPro(O_{1F},O_1))),
Π_{Cntxt_2}(semCompare(Cntxt_2,semPro(O_{2F},O_2))))
= <ANSWER,{C_1,C_2},{O_1,O_2},{ren_{O_1}(C_1,A_1),ren_{O_2}(C_2,A_2)},M>
M $\equiv$ ANSWER=OJoin(g(A_1,A_2),OSelect(p',O_1),OSelect(p",O_2))

A.5 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 multi-database 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
Union Compatibility Conflicts		β	β	β	β	α
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 2: 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.

Note : In [SK92] we have identified and defined the above schematic conflicts. We have, using the concept of **semantic proximity** identified the possible semantic similarities between two objects having structural conflicts. However, in this paper, we represent the structural similarity between objects having schematic conflicts and some semantic similarity using the concept of **schema correspondences**. The schema correspondence(s) are then associated with and as component(s) of the semantic proximity.