Schema Correspondences between Objects with Semantic Proximity

Vipul Kashyap[1] and Amit Sheth[2]

[1]Department of Computer Science, Rutgers University,
New Brunswick, NJ 08903

[2]Bellcore, 444 Hoes Lane, Piscataway, NJ 08854-4182

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 introduce a uniform formalism called *schema correspondences* to represent structural similarities between the objects. We represent the semantic similarities between the objects using the concept of *semantic proximity*. We show how the reconciliation is achieved by illustrating the association of the schema correspondence(s) with and as component(s) of the semantic proximity. We also provide a data model independent *semantic taxonomy* on the basis of the semantic proximity defined. We then enumerate and classify the *schematic and data conflicts*. The association between the schema correspondences and semantic proximity helps represent the *possible semantic similarities* between two objects having these conflicts. One representation of *uncertain information* using semantic proximity as the basis is explored. Issues of *inconsistent information* are also discussed in the framework of semantic proximity.

1 Introduction

Many organizations face the challenge of interoperating among multiple independently developed database systems to perform critical functions. With high interconnectivity and access to many information sources, the primary issue in the future will not be how to efficiently process the data that is known to be relevant, but 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

1

related objects. In this paper, we are interested in the **dual perspective** that emphasizes both the semantic similarities and the schematic (representational/ structural) differences.

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 a comprehensive taxonomy of schematic conflicts which subsumes most of the taxonomies found in literature (refer to the table in the Appendix). Attempts have been made to represent structural similarities in [CHS91, SPD92, UW91, GB91]. 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 in the following manner :

- We represent the structural similarity between objects having schematic conflicts and some semantic similarity using the concept of **schema correspondences** which is proposed in Section 2 of this paper. We characterize the degree of semantic similarity between a pair of objects using the concept of **semantic proximity** [SK92] which is proposed in Section 3 of this paper. The schema correspondence(s) are then associated with and as component(s) of the semantic proximity.

- We develop a **semantic taxonomy** emphasizing semantic similarities between objects and show its relationship to a **structural taxonomy** emphasizing schematic (structural/representational) differences among the objects [SK92]. The association between the schema correspondences and semantic proximity helps represent the possible semantic similarities between two objects having these conflicts.

Understanding and representing semantic similarities and schematic differences between objects may involve understanding and modeling **uncertainty**, **inconsistency** and **incompleteness** of information pertaining to the objects (at both intensional and extensional levels), and the relationships between the objects. We address some of the issues of uncertainty and inconsistency. In Section 4, we describe *fuzzy terminological relationships* [FKN91] by expressing the fuzzy strengths as a function of the semantic proximity between two objects. Section 7 addresses the *data value incompatibility problem* which arises out of the inconsistency between related data and the semantic similarities possible between inconsistent data.

The remaining sections deal with a broad class of schematic differences. We first define the schema correspondences between the objects having those differences. They are associated with and are a part of the definition of the semantic proximity between the objects. Section 5 deals with the *domain incompatibility problem* [CRE87] which arises when attributes have different domain definitions. Section 6 discusses the *entity definition incompatibility problem* [CRE87] which arises when the entity descriptors used for the same entity are partially compatible. Section 8 deals with the *abstraction level incompatibility problem* [DH84] which arises when the same entity is represented at different levels of abstraction. Section 9 deals with the *schematic discrepancy problem* [KLK91] which arises when data in one database corresponds to schema elements in another.
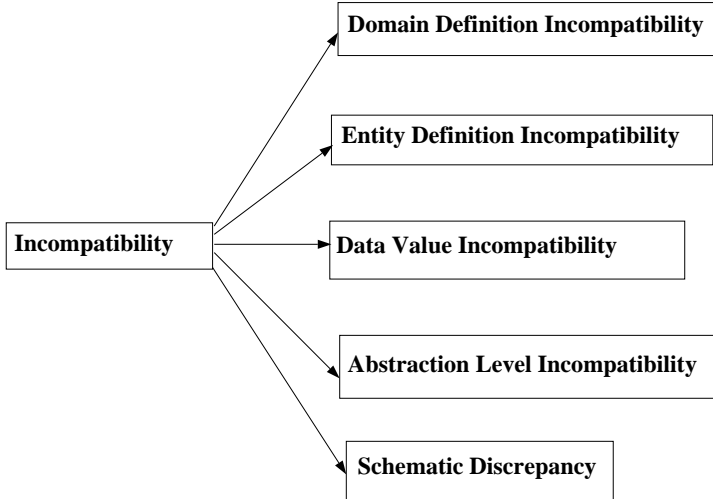
Figure 1: Structural Incompatibilities due to Heterogeneity

2 Semantic Similarities between objects

In this section, we discuss the concept of *semantic proximity* to characterize semantic similarities between objects, and use it to provide a classification of semantic similarities between objects. We also illustrate in Section 5, how the semantic proximity so defined can provide a basis for assigning fuzzy strengths to model uncertainty in the similarities between objects.

We distinguish between the *real world,* and the *model world* which is a representation of the real world. 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.*[1]

Wood [Woo85] defines semantics to be "the scientific study of the relations between signs and symbols and what they denote or mean." 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].

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 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

---

[1]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].

[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.
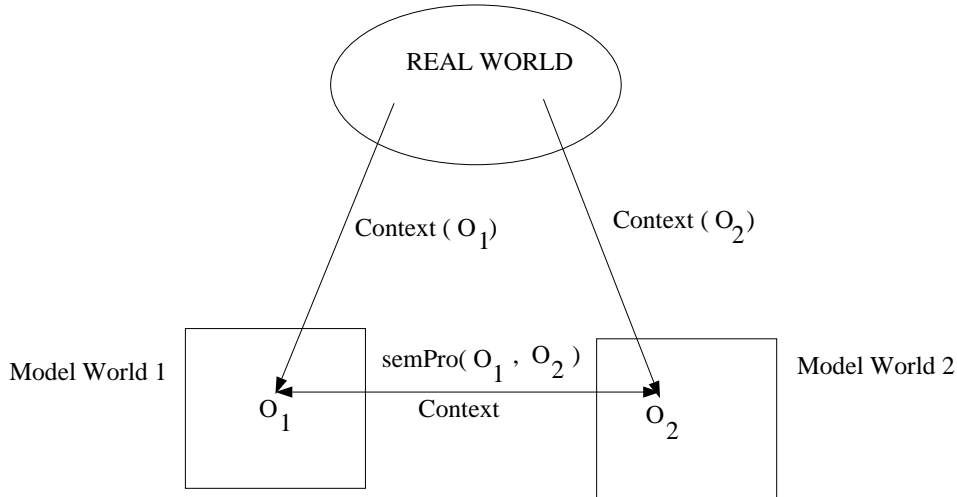
3

Figure 2: Semantic Proximity between two Objects

is *semantically closer* than semantic relationship, and so on.

2.1 A model for Semantic Classification

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.

A context of an object is the primary vehicle to capture the RWS of the object. Thus, the respective contexts of the objects, and to a lesser extent the abstraction used to map the domains of the objects, help to capture the semantic aspect of the relationship between the two objects.

2.2 Context(s) of the two Objects : 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. Some of the alternatives for representing a context in a *multidatabase system* are as follows.

- In [SM91], the context is identified as the semantics associated with an application's view of existing data and is called the **application semantic view**. They propose a rule-based representation to associate metadata with a given attribute, and use this rule based representation to define the application's semantic view of the data.

- Just as a context may be associated with an application, it can also be associated with a **database** or a group of databases (e.g., the object is defined in the context

4

of DB1).

- When many entities participate in a relationship, the entities can be thought of as belonging to the same context, which in this case is identified as the **relationship** in which the entities participate.

- In a federated database approach, we can use a **federated schema** [SL90] to identify a context to which two objects may belong to.

- From the five-level schema architecture for a federated database system [SL90], a context can be specified in terms of an **export schemas** (a context that is closer to a database) or an **external schema** (a context that is closer to an application). We can also build a context hierarchy, by considering the contexts associated with the external schemas to be subcontexts of the context associated with the appropriate federated schema.

- At a very elementary level, a context can be thought of as a **named collection** of the domains of the Objects.

- When using a well defined ontology, such as Cyc [Guh90], a well defined partition (called Microtheory) of the ontology can be assigned a context.

- 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.

Additional research is needed to identify appropriate representations of context, and develop a practical framework for semi-automatic ways of comparing and manipulating contexts (e.g., taking a union of two contexts). While it may not be possible to precisely define the context of an object, it may be useful to simply name it at a specific level of information modeling architecture (e.g., external schema or federated schema). A partial context specification can be used by humans to decide whether the context for modeling of two objects is the same or different, and whether the comparison of semantic similarity of objects is valid in all possible contexts or specific ones. Examples and discussions in the rest of the paper will clarify these points. A more detailed discussion of the nature of context and it's relation to semantics is provided in [SK93].

2.3 The Structural Components

2.3.1 Abstraction used to map the objects

We use the term abstraction to refer to a mechanism used to map the domains of the objects to each other or to the domain of a common third object. In the Section 3, we define the concept of schema correspondences to express the abstractions in a uniform formalism.

It must be noted that an abstraction by itself cannot capture semantic similarity, as it is always possible to construct a mapping between two semantically unrelated objects

5

D1 is a subset of D2 x D3 x D4

Domain of Object(D1)

X   X

Domain of attr(D2)   **Domain of attr(D4)**
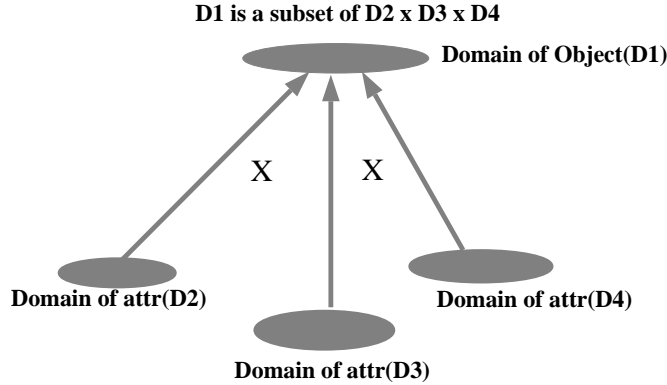
Domain of attr(D3)

Figure 3: Domain of an Object and it's Attributes

[SG89]. However, if there is a semantic similarity between two objects, then we should be able to do so *wrt* a particular (or all) context(s). Thus we associate the abstraction with and as component(s) of semantic proximity.

2.3.2 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. 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.

2.3.3 States (extensions) of the objects

The state of an object can be thought of as an extension of an object recorded in one or more databases. However, this extension must not be confused with the actual state of the entity (according to the Real World Semantics) being modeled. Two objects having different extensions can have the same state Real World Semantics (and hence be semantically equivalent). See section 8 for a more detailed discussion.

3 Uniform formalism for representation of Object Correspondences

In this section we are interested in the representation of structural similarities between two objects which have some semantic affinity independent of data model considerations.

Here we propose a uniform formalism for representation of object correspondences called **schema correspondences**. It is a generalization of the connectors proposed in [CRE87]. This uniform formalism may involve mappings between the domains of these objects. The key aspect of our approach is that the schema correspondences so defined are associated with and are components of the semantic proximity defined in the previous section. They maybe associated with the context(s) (all or a particular one) in which the semantic proximity is defined.

An alternative viewpoint would be to view schema correspondences as a projection of the semantic proximity from the semantic space to the structural space *wrt* the context. Thus, a point in the structural space (structural schema correspondence) can correspond to several points in the semantic space (semantic proximity). This is illustrated by an example in section 4.5.2.

3.1   Previous approaches to represent Structural Similarities

In the Carnot project [CHS91] during the schema representation phase, the schema of an information resource is represented in the formalism of the global schema. The mapping between each individual information resource and global schema 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 [LG90]. Here we are interested in the structural correspondences between two objects, and an extensive ontology as in [LG90] may not be required. Also identifying an object as a relation/entity/class may not be necessary as we are interested in a data-model independent manner of representing structural object correspondences.

There have been various model describing techniques to represent object correspondences [GB91, UW91]. One approach has been to use a higher order language like M(DM)-L [GB91] to define metatypes which formalize the syntax and semantics of a data model construct by a collection of second order logic formulae. Two types are equivalent if the corresponding sets of second-order formulae are logically equivalent. Since we are interested in specifying the correspondences between two objects at the schema level, we may not need the full scope of a second-order logic.

Another model describing approach [UW91] has been to use a semantic data model to represent its own structural semantics as well as the semantics of the other data models in a multidatabase system. Since we are identifying the entities of the different component databases as objects and are interested in the resulting schema correspondences between them, maintaining the mappings between the various data-model constructs (viz. Class in an OODB maps to a relation in a RDB) may not be necessary.

An alternative approach [SPD92] is to model commonalities between the schemas of the various component databases as *correspondence assertions* which state some relationship between two elements and their instances in the two databases. The assertions can be of various types, viz. between elements of the same type, between elements of different types and between links. However these assertions do not give us the mappings between the various elements involved in the assertions. They are also not able to capture the correspondences between the *values* of a schema element in one database and the schema elements in other databases [DAODT85, KLK91].

3.2 Schema Correspondences as a uniform formalism

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

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

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

- O_1 and O_2 are objects in the model world. They are representations or intensional definitions in the model world (e.g., an object class definition in object-oriented models).

- 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 entity modeled by O_i. If O_i models objects at the attribute level, then attr(O_i) is an empty set.

- Ψ is a mapping (possibly higher-order) expressing the correspondences between objects, their attributes and the values of the objects/attributes.

3.3 Abstraction used to map the objects

We use the term abstraction to refer to a mechanism used to map the domains of the objects to each other or to the domain of a common third object. The concept of dynamic attributes has been proposed in [LA86] to specify the mappings between different attributes. Various ways of implementing the mappings are proposed (viz. mathematical formulae, tables, programs). However, we here focus on the specification of mappings at a conceptual level between the domains of attributes and objects. Some of the more useful and well defined abstractions are enumerated below. We shall represent the object correspondences in the uniform formalism of schema correspondences as proposed in the previous section.

- **A total 1-1 value mapping** between the domains of the objects, i.e., for every value in the domain of one object, there exists a value in the domain of the other object and vice versa. Also there is a one to one correspondence between the values of the two domains.

  Example : Consider two entities EMPLOYEE and WORKER in two different databases defined as follows :

  ```
  EMPLOYEE(EmpNo, Name, Addr)
  WORKER(SSNo, FName, LName, Addr)
  ```

CITIES	COUNTRIES
New York, Los Angeles, Washington D.C.	United States of America
Bombay, Delhi, Calcutta, Madras	India
London, Manchester, Essex	Great Britain
Montreal, Vancouver, Edmonton	Canada
Munich, Bonn, Berlin	Germany

Table 1: Mapping between Cities and Countries

The correspondence between the two can be represented as :

<EMPLOYEE, {EmpNo, Name}, WORKER, {SSNo, FName, LName}, Ψ >
where Ψ is a total 1-1 value mapping as follows :
Ψ : EmpNo $\times$ Name $\leftrightarrow$ SSNo $\times$ FName $\times$ LName

- **A partial many-one mapping** between the domains of the objects. In this case some values in either domain might remain unmapped (i.e. partial mapping), or a value in one domain might be associated with many values in another domain (i.e. many one mapping).

  Example : The correspondence between the CITIES and the COUNTRIES they belong to can be represented as :

  <CITIES, ϕ, COUNTRIES, ϕ^3, Ψ >
  where Ψ is a many-one mapping defined in Table 1.

- The **generalization** abstraction to relate the domains of the concerned objects. One domain can generalize/specialize the other, or domains of both the objects can be generalized/specialized to a third domain. Both can be expressed using the mechanism of mappings between the domains of the concerned objects as follows :

  - Generalization can be expressed as a total, many-one mapping from the union of the domains of the objects being generalized to the domain of the generalized object.

  - Specialization can be expressed as a total, many-one mapping from the domain of the specialized object to the domain of the object being specialized.

  Example : The following example illustrates how one can represent the generalization abstraction in a uniform formalism. The specialization abstraction can be represented analogously. Consider two entities in different databases STUDENT, PART-TIMER and GRAD-STUDENT.

---

[3]Since CITIES and COUNTRIES are attributes we assume that the set of attributes attached to them is empty. However, these fields might not be empty for composite or aggregate attributes.

```
PART-TIMER(Id#, Name, Major, Credits)
GRAD-STUDENT(SS#, Name, Major, Advisor)
STUDENT(Id#, Name, Major)
```

The correspondence between these three entities can be represented as :

<STUDENT, {Id#}, {PART-TIMER, GRAD-STUDENT}, {Id#, SS#}, Ψ >
where Ψ is a many-one mapping as follows :
Ψ : PART-TIMER.Id# $\cup$ GRAD-STUDENT.SS# $\rightarrow$ STUDENT.Id#

- The **aggregation** abstraction to relate the domains of the objects. This can be expressed as a partial, 1-1 or many-one mapping between the cross-product of the domains of the objects being aggregated and the domain of the aggregated object.

  Example : Consider two entities in different databases PLAYER and TEAM. A particular TEAM entity is a set of many PLAYER entities and may be considered an aggregation of PLAYER.

```
TEAM(Id#, AvgScore, Hometown)
PLAYER(Id#, Score, Hometown)
```

The correspondence between these entities can be represented as :

<PLAYER, {Id#, Score}, TEAM, {Id#, AvgScore}, Ψ >
where Ψ is a many-one mapping as follows :
Ψ : PLAYER.Id# $\times$ Score $\rightarrow$ TEAM.Id# $\times$ AvgScore

- **Functional Dependencies**. They can be expressed as a partial, many-one mapping between the cross-products of the domains of the determining objects and the cross-product of the domains of the determined objects.

4   A Semantic Taxonomy

We use the **semantic proximity** and the **schema correspondence** defined in the previous sections to propose a semantic taxonomy of the various types of semantic similarities between the objects. As indicated earlier, we define qualitative measures to distinguish between the terms introduced in [She91b], *viz. semantic equivalence, semantic relationship, semantic relevance* and *semantic resemblance*. This depends on the values taken by context and the type of schema correspondence associated with that value of context as a part of the semantic proximity.

In Sections 6-10 we show the relationship of the semantic taxonomy to a structural taxonomy (which emphasizes the structural differences) by identifying the possible semantic similarities between two objects having structural differences.

10

4.1 The role of context in semantic classification

In our classification scheme, we are interested in the cases where the context(s) of the objects under consideration can be determined to be one of the following. In cases other than ALL and NONE, specific instances of semPro must name context(s) explicitly.

- ALL, i.e., the semPro of the objects is being defined *wrt* all known contexts. The specific context need not be named.

- SAME, i.e., the semPro of the objects is being defined *wrt* the same context. The context must be explicitly specified in an instance of a semPro.

- SOME, i.e., the semPro of the objects are being defined *wrt* more than one context. The applicable contexts must be individually or collectively specified in an instance of a semPro.

- SUB-CONTEXTS, when the semPro can be defined in a previously defined context that is further constrained. The subcontext must be specified in an instance of a semPro.

- NONE, i.e. the objects under consideration do not exhibit any useful semantic similarity under any known context.

4.2 Semantic Equivalence

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

semPro(O_1, O_2) = <ALL, Abstraction, (D_1, D_2), _>[4]
where Abstraction is expressed in the following schema correspondence :

< O_1, {A_1,...,A_n}, O_2, {B_1,...,B_n}, Ψ >
where Ψ is a total 1-1 value mapping defined as follows :
$\Psi : A_1 \times \ldots \times A_n \leftrightarrow B_1 \times \ldots \times B_n$

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

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

---

[4]"_" stands for don't care.

where Abstraction is expressed in the following schema correspondences :

$< O_1, \_, O_2, \_, \Psi >$
where Ψ is a total 1-1 value mapping given as follows :
$\Psi : (D_1, S_1) \leftrightarrow (D_2, S_2)$.

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

4.3   Semantic Relationship

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

semPro(O_1, O_2) = $<$ALL, Abstraction, (D_1, D_2), $\_>$
where Abstraction is expressed in the following schema correspondence :

$< O_1, attr(O_1), O_2, attr(O_2), \Psi >$
where Ψ may be a partial many-one mapping, generalization mapping or aggregation mapping between O_1 and O_2 depending on the abstraction.

4.4   Semantic Relevance

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

semPro(O_1, O_2) = $<$SAME, ANY, (D_1, D_2), $\_>$

4.5   Semantic Resemblance

This is the weakest measure of semantic proximity, which might be useful in certain cases. Here, we consider the case where the domains of two objects cannot be related to each other by any abstraction in any context. Hence, the exact nature of semantic proximity between two objects is very difficult to specify. In this case, the user may be presented with extensions of both the objects. In order to express this type of semantic similarity, we introduce an aspect of context, which we call **role**, by extending the concept of role defined in [EN89]. Semantic resemblance is defined in detail in section 4.5.2.

4.5.1 Role played by an object in a context

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

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

The mapping defined above may be multi-valued, as it is possible for an object to have multiple roles in the same context. However, for our purposes, we shall assume the mapping to be a single-valued binary function.

4.5.2 Roles and Semantic Resemblance

Whenever two objects cannot be related to each other by any abstraction in any context, but they have the same roles in their respective context(s) (where the respective contexts may or may not be the same), they are said to *semantically resemble* each other. This is a generalization of DOMAIN-DISJOINT-ROLE-EQUAL concept in [LNE89].

$semPro(O_1, O_2) = <context, NONE, (D_1, D_2), _>$
where $context = context(O_1) \cup context(O_2)$
and $D_1 \neq D_2$
and $role\text{-}of(O_1, context) = role\text{-}of(O_2, context)$

Example : In this example we demonstrate the semantic aspect of the similarity between two objects captured by a context. Consider two objects O_1 (TELECOM-EMPLOYEE) and O_2 (BANK-EMPLOYEE) as defined below.

We show how it is possible for two objects to be semantically closer in one context than in another context. Thus, it is possible for the same structural schema correspondence to be associated with and a part of different semantic proximities.

```
TELECOM-EMPLOYEE(ID, SALARY, ...)
BANK-EMPLOYEE(ID, SALARY, ...)
```

Suppose the IRS (a government income tax department) wants to query both these objects wrt the tax bracket both types of employees fall in. O_1 and O_2 can be defined to be semantically relevant using the following information :

$context = context(O_1) = context(O_2) = IRS$ (i.e., $context = SAME$)
$abstraction : EMPLOYEE(ID, SALARY) = generalize(O_1, O_2)$

What if there is no single context, such as the one needed for the IRS application, in which the above objects are to be considered ? Should the objects then, be considered for schema integration ? The weaker semantic proximity of semantic resemblance can be defined between O_1 and O_2 using the following information :
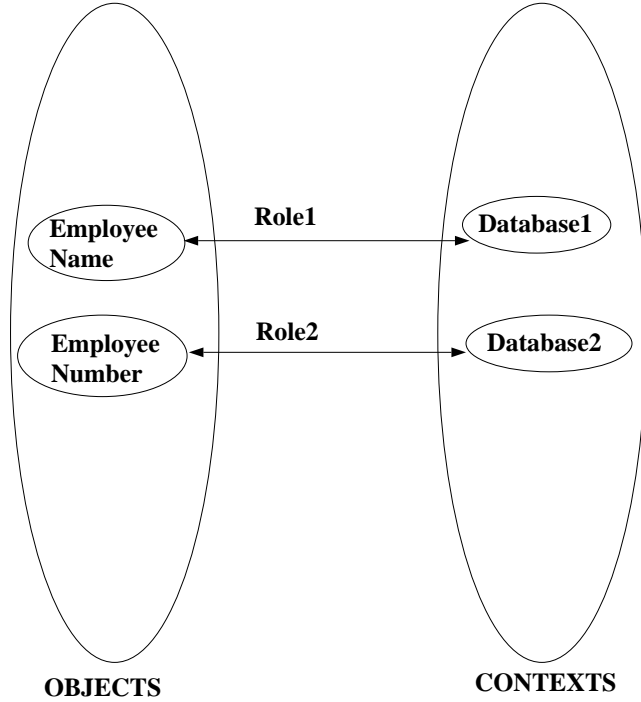
Role1 = role-of(EmployeeName, Database1) = Identifier
Role2 = role-of(EmployeeNumber, Database2) = Identifier
Thus, ¬Role1 = Role2

Figure 4: Roles played by objects in their contexts

$$\text{context}(O_1) = \text{TELECOMMUNICATION}$$
$$\text{context}(O_2) = \text{BANKING}$$
$$\text{role-of}(O_1, \text{TELECOMMUNICATION}) = \text{SUBORDINATE}$$
$$\text{role-of}(O_2, \text{BANKING}) = \text{SUBORDINATE}$$

4.6   Semantic Incompatibility

While all the 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 there is no context and no abstraction in which the domains of the two objects can be related. Furthermore, the two objects cannot have similar roles in the context(s) in which they exist.

$\text{semPro}(O_1, O_2) = <\text{NONE, NEG}, (D_1, D_2), \_>$
where context = $\text{context}(O_1) \cup \text{context}(O_2)$ is undefined,
and   Abstraction = NEG, signifying the dissimilarity
and D_1 may or may not be equal to D_2
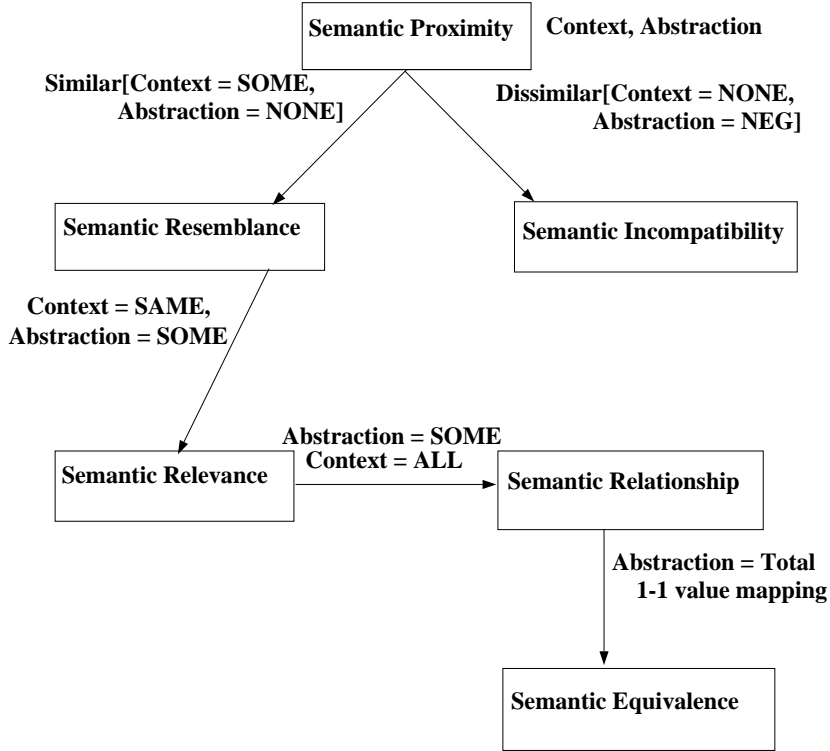and   role-of(O_1, context) and role-of(O_2, context) are incomparable

14

Figure 5: Semantic classification of object similarities

5   Semantic Proximity and Uncertainty Modeling

Specifying object relationships involve determining equivalence or subtype assertions between schema objects (e.g., entities) and between attributes. One approach has been to group attributes in a taxonomy of equivalence classes [ELN86, ME84] or a subtype hierarchy [SG89][SM92] and specify object relationships based on assertions among attributes. Another approach has been to annotate attributes or objects with a set of concepts from a global concept space [YSDK91] and determine the object relationships based on the concepts they are related to. Whether giving assertions among attributes or concepts, in practice we often can give only a fuzzy (i.e. uncertain or ambiguous) assertion. Modeling uncertainty can help in identifying a larger class of assertions leading to better identification of semantic proximities among the objects.

5.1   Previous approaches to model uncertainty

Several approaches have been proposed to model uncertainty in the relationships between objects. One approach has been to determine the similarity of objects using fuzzy and incomplete terminological knowledge [FKN91] together with schema knowledge. The difficulty of this approach is that the assignment of fuzzy strengths is based on intuition, and albeit arbitrary. We are of the opinion that such an assignment of certainty measures is a context sensitive process and depends on the relation between the domains of the terminological entities involved. Thus, these factors could form a basis for assignment of

the fuzzy strengths.

Another approach has been that of using partial values and maybe tuples [DeM89] using the framework of 3-valued logic. In this approach, the partial and maybe information has a more formal basis, i.e., a value mapping between the domains of the objects. In our opinion fuzzy logic gives us a more extensive framework than 3-valued logic with which to represent the full range of uncertain information. Mapping information as a basis for determining uncertainty is inadequate in many cases and in such cases, using the context and the extensions of the objects can be helpful.

Example : Consider two objects STUDENT and DEPARTMENT defined as follows,

```
STUDENT(Id#, Name, Grade)
DEPARTMENT(Num, Name, Address)
```

Let Domain(STUDENT.Id#) = {123, 456, 789}
and Domain(DEPARTMENT.Num) = {321, 654, 987}

It is possible to define a Mapping between the domains defined above, but this does not mean that STUDENT.Id# is equivalent to DEPARTMENT.Num

A third approach is to use discrete probability distributions to model uncertainty [BGMP90, TCY93]. However, probability values are either assigned as a measure of belief or by an analysis of the underlying sample. If the values are assigned as a measure of belief as in [Zad78], then it is necessary to specify an underlying basis for specifying these measures. Also, if these values are assigned by analyzing the underlying sample, then they depend on the extensions or state of the objects, which might be rendered obsolete in a continually changing database. The *implicit independence assumption* which says that the probabilities modeling the uncertainty of an attribute are independent of the probability values of the other attributes also does not appear to accurately reflect the Real World Semantics.

Example : Consider two objects INSTR1 and INSTR2 in two different databases.

```
INSTR1(SS#, HPhone, OPhone)
INSTR2(SS#, Phone)
```

M_1 : INSTR1.HPhone $\rightarrow$ INSTR2.Phone
M_2 : INSTR1.OPhone $\rightarrow$ INSTR2.Phone
be two mappings define between the attributes of the objects.

Obviously M_1 and M_2 are not independent of each other and are related to each other through the following schema correspondences :

$<$ INSTR1, {SS#, HPhone}, INSTR2, {SS#, Phone}, Ψ_1 $>$
where Ψ_1 could be a partial many-one value mapping,
Ψ_1 : INSTR1.SS# $\times$ INSTR1.HPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone

< INSTR1, {SS#, OPhone}, INSTR2, {SS#, Phone}, Ψ_2 >
where Ψ_2 could be a partial many-one value mapping,
Ψ_2 : INSTR1.SS# $\times$ INSTR1.OPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone

We propose representing the uncertainty in the integration assertions by using the concept of *semantic proximity* defined earlier. We believe that the semantic proximities can provide a well defined basis for the assignment of fuzzy strengths. We give in section 5.2, an explanation of how this may be achieved. In section 5.3, we also demonstrate the simulation of heuristics used to assign the fuzzy strengths.

5.2   Fuzzy Strengths as a function of Semantic Proximity

In this section we establish the semantic proximity as a basis for the assignment of fuzzy strengths to the terminological relationships between two semantically similar objects. As noted in the previous section, when we assign fuzzy strengths to semantic similarities between schema objects, they should reflect the Real World Semantics. Thus any such assignment of belief measures should depend on and reflect the following.

- The **context(s)** to which the two schema objects belong to.

- The **mapping(s)** which may exist between the domains of the objects or the domains of the individual attributes of the objects. Here, it may be noted that the mappings between two attributes of the objects might not be independent of each other, but maybe dependent. Thus, instead of having mappings $A_{1,1} \rightarrow A_{2,1}$ and $A_{1,2} \rightarrow A_{2,2}$, where $A_{i,j}$ is the jth attribute of the ith object, we might have mappings between pairs of attributes, i.e. $A_{1,1} \times A_{1,2} \rightarrow A_{2,1} \times A_{2,2}$. Hence, the implicit independence assumption of [BGMP90] might not accurately reflect the mappings.

- The **state(s)** or the extensions of the two objects.

The semantic proximity described in the previous section is able to capture this information which represents the semantic similarity between two objects according to the Real World Semantics. Also the interactions between any two attributes of an object can be captured using the interactions between the mappings of the two attributes, thus avoiding the need for the implicit independence assumption.

We define an **uncertainty function** μ between two objects O_1 and O_2 which maps the semantic proximity to the real interval [0,1]. Thus

μ : semPro(O_1, O_2) $\rightarrow$ [0,1]
i.e., μ(Context, Abstraction, (D_1, D_2), (S_1, S_2)) = X where $0 \leq X \leq 1$.

μ is a **user defined** function such that it accurately reflects the Real World Semantics and may not have specific mathematical properties. It may or may not be a computable function. If it is a computable function, that would mean that we can automate the process of assigning the fuzzy strengths to the semantic relations between schema objects. However, it would require the semantic proximities discussed earlier. Two users might

choose to define the function differently, but now we have a basis on which to judge, which function is a better reflection of the Real World Semantics. If μ is not computable, a human makes an assignment based on the context(s), the mapping(s) between the domains of the two objects, and possibly the states of the two objects.

Earlier we defined the various kinds of semantic proximities. Now, based on these semantic proximities, we develop a *bounded correctness criterion* which any user defined uncertainty function may follow.

Bounded correctness criterion

Given a user defined uncertainty function μ, let the values to which it maps the various semantic proximities be given as follows :

$\mu(\text{State-Equivalent}) = X_{StateEq}$
$\mu(\text{Domain-Equivalent}) = X_{DomEq}$
$\mu(\text{Related}) = X_{Relat}$
$\mu(\text{Relevant}) = X_{Relev}$
$\mu(\text{Resemble}) = X_{Res}$
$\mu(\text{Incompatible}) = X_{In}$

The bounded correctness criterion that a heuristic may meet to justify consistent derivation of the fuzzy strengths is specified as follows :

1. $X_{StateEq} = 1$

2. $0 < X_{Res} \leq X_{Relev} \leq X_{Relat} \leq X_{DomEq} < 1$

3. $X_{In} = 0$

5.3 Simulation of heuristics using semantic proximity for assignment of Fuzzy Strengths

In this section we discuss whether the definition of the user defined uncertainty function μ described in the previous section can capture the heuristics used to assign fuzzy measures. We show how some of the proposed heuristics used for assignment of fuzzy strengths to the relationships can be simulated using the semantic proximities and by defining an appropriate uncertainty function.

5.3.1 The heuristic of % of common attributes

This is a very simple heuristic [ELN86] to represent using the semantic proximity. It is a heuristic which essentially exploits the *structural similarity* between two entities. The uncertainty function will be independent of the context(s) and the states of the objects. Given the semantic proximity, the uncertainty function μ can be defined as follows :

$\mu(\text{Contexts, Abstraction, } (D_1, D_2), (S_1, S_2)) = \frac{|\{attr(O_1)\} \cap \{attr(O_2)\}| \times 100}{|\{attr(O_1)\}|}$

where the schema correspondences (between attributes A_i of object O_1 and A_j of object O_2) may be represented as follows:

$< O_1, \{A_i\}, O_2, \{A_j\}, \Psi_i >$
where Ψ_i is a total 1-1 value mapping as follows :
$\Psi_i : A_i \leftrightarrow A_j$

Example : Consider two Union Incompatible entities as follows :

```
Student1(Id#, Name, Grade)
Student2(Id#, Name, Address)
```

The semantic proximity can be given as :
semPro(Student1, Student2) = $<$ALL, Abstraction, (D_1, D_2), _$>$
where Abstraction is expressed in the following schema correspondence :

$< \text{Student}_1, \{\text{Id\#, Name}\}, \text{Student}_2, \{\text{Id\#, Name}\}, \Psi >$
where Ψ may be a total 1-1 value mapping,
$\Psi : \text{Student}_1.\text{Id\#} \times \text{Student}_1.\text{Name} \leftrightarrow \text{Student}_2.\text{Id\#} \times \text{Student}_2.\text{Name}$

The uncertainty function is then given as :
$\mu(\text{Student}_1, \text{Student}_2) = \frac{|\{Id\#, Name\}| \times 100}{|\{Id\#, Name, Grade\}|}$

5.3.2 The heuristic of instance participation

This heuristic uses the concept of the cardinality constraints of the entities participating in the mappings [EN89, VH91] to define the uncertainty function. Although this function expresses more semantic information than the previous one, it is independent of the context(s) of the two objects. Using the concept of cardinality constraints, we can define the uncertainty function as follows.

Let O_1 and O_2 be two schema objects and let their semantic proximity be,

semPro(O_1, O_2) = $<$ALL, Abstraction, (D_1, D_2), _$>$
where Abstraction is a total many-one value mapping between the domains with the cardinality constraints of the domains participating in the mapping given as :
$D_1 \rightarrow (min_1, max_1)$ and $D_2 \rightarrow (min_2, max_2)$
where min_i and max_i are the minimum and maximum number of elements of domain D_i participating in the mappings.

$\mu(\text{Contexts, Abstraction}, (D_1, D_2), (S_1, S_2)) = \frac{(min_1 + max_1)(min_2 + max_2)}{4 \times max_1 \times max_2}$

6 Domain Incompatibility Problem

In this section we discuss the incompatibilities that arise between two objects when they have differing definitions of semantically similar attribute domains. A broad definition of this incompatibility was given in [CRE87]. We examine in detail the aspects in which two attribute domain definitions can differ and give a comprehensive enumeration of the resulting types of incompatibilities. Some of these aspects have been identified in [DAODT85]. For each enumerated conflict, we specify the schema correspondence in association with and as a component of the semantic proximity.

6.1 Naming Conflicts

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

Example : Consider the two entities STUDENT and GRADUATE in different databases as follows :

```
        STUDENT(Id#, Name, Address)
        GRADUATE(SS#, Name, Address)
STUDENT.Id# and GRADUATE.SS# are synonyms.
```

The schema correspondence between these two objects can be given as :

$< $ STUDENT, $\{$Id$\#\}$, GRADUATE, $\{$SS$\#\}$, $\Psi >$
where Ψ is a total 1-1 value mapping defined as,
$\Psi : \text{Id}\# \leftrightarrow \text{SS}\#$

Mappings between synonyms can often be established *wrt* all contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent*.

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

Example : Consider the two entities STUDENT and BOOK in different databases as follows :

```
        STUDENT(Id#, Name, Address)
        BOOK(Id#, Name, Author)
STUDENT.Id# and BOOK.Id# are homonyms.
```

Since homonyms are semantically unrelated, there cannot be any context in which there is an abstraction that maps one homonym to another. In such cases, two objects O_1 and O_2 can be considered to be *semantically incompatible*.

6.2 Data Representation Conflicts

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

Example : Referring to the second example in section 6.1,

```
STUDENT.Id# is defined as a 9 digit integer.
GRADUATE.SS# is defined as an 11 character string.
```

Conversion mappings or routines between different data representations can often be established *wrt* all contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent*.

The schema correspondence between these two objects can be given as :

$< $ STUDENT, {Id#}, GRADUATE, {SS#}, $\Psi >$
where Ψ is a total 1-1 value mapping defined as,
Ψ : Id# $\leftrightarrow$ SS#
For instance, SS# $= \Psi($Id#$) \equiv \Psi(123456789) = $ "123-45-6789"

6.3 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 $(SALARY_1)$ and £$(SALARY_2)$.

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 contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent*.

The schema correspondence between these two objects can be given as :

$< $ SALARY$_1$, ϕ, SALARY$_2$, ϕ, $\Psi >$
where Ψ is a total 1-1 value mapping defined as,
Ψ : SALARY$_1$ $\leftrightarrow$ SALARY$_2$
In this case SALARY$_2$ $= \Psi($SALARY$_1) = 2 \times$ SALARY$_1$

6.4 Data Precision Conflicts

Two attributes that are semantically similar might be represented using different precisions [DAODT85]. This case is different from the previous case in that 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 :

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

Table 2: Mapping between Marks and Grades

```
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 contexts. The other way round, e.g., given a letter grade identifying the precise numerical score, is typically not possible. In such cases, two objects O_1 and O_2 can be considered to have a *semantic relationship*.

The schema correspondence between the two objects can be represented as :

$< MARKS, \phi, GRADES, \phi, \Psi >$
where Ψ is a total many-one value mapping as defined in Table 2.

6.5 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. These conflicts were discussed in [KS91] and can be classified as the broader class of domain incompatibility conflicts. In this case, two attributes might have different default values in different databases. 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 contexts. However, it is often possible to define a mapping between them *wrt* the same context. In such cases, the two objects O_1 and O_2 can be considered to be *semantically relevant*, i.e., their *semantic proximity* can be defined as follows :

semPro(Age_1, Age_2) = $<$SAME, Abstraction, (D_1, D_2), _$>$
Context = SAME = LegalDriver for Age_1 and Age_2.
The Abstraction may be expressed by the following schema correspondence :

$< Age_1, \phi, Age_2, \phi, \Psi >$
where Ψ is a 1-1 mapping between two singleton sets.

22

$\Psi : \{18 \text{ yrs}\} \leftrightarrow \{21 \text{ yrs}\}^5$

6.6 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 : Consider two constraints C1 and C2 in two different databases on the same attribute.

```
C1 : Age < 18
C2 : Age > 21
C1 and C2 are inconsistent and hence the integrity constraints on
the attribute Age conflict.
```

Depending on the nature of the integrity constraints involved, it might be possible to generalize the constraints and have a mapping from the specific to the general constraints. However, in certain cases the nature of inconsistency might be such that a mapping might not be possible. Even in that case, the objects O_1 and O_2 can be considered to *semantically resemble* each other, if they have the same role in their respective context(s).

semPro(Age_1, Age_2) = <context, NONE, (D_1, D_2), _>
where context = context(Age_1) $\cup$ context(Age_2)
and $D_1 \neq D_2$
and   role-of(Age_1, context) = role-of(Age_2, context) = AGE

7 Entity Definition Incompatibility Problem

In this section we discuss the incompatibilities that arise between entities of the same type when the entity descriptors used by the objects are only partially compatible. A broad definition of this class of conflicts was given in [CRE87]. Here we examine in detail the scenarios in which the entity definitions of semantically similar entities might conflict to give a more precise and comprehensive enumeration of this class of conflicts. For each enumerated conflict, we specify the schema correspondence in association with and as a component of the semantic proximity.

7.1 Database Identifier Conflicts

In this case, the entity descriptions in two databases are incompatible because they use identifier records that are semantically different. In the case of the relational model, this would translate to two relations modeling the same entity having semantically different keys, and is also known as the *key equivalence problem*.

---

[5]It should be noted however that this schema correspondence and the component mapping is associated with the context "LegalDriver".
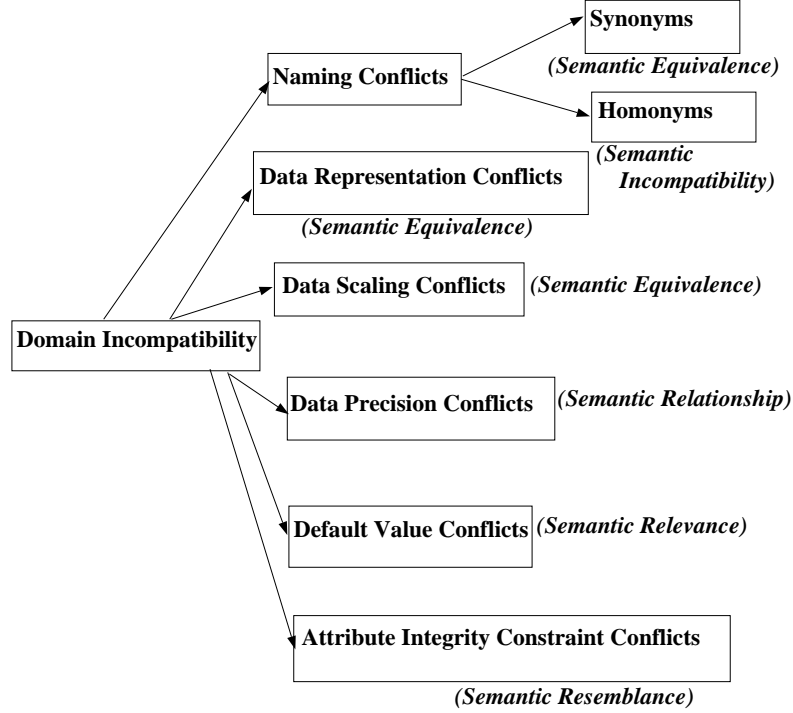
Figure 6: Domain Incompatibility and the *likely* types of semantic proximities

Example : Consider two entities STUDENT1 and STUDENT2 in different different databases as follows :

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

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 proximity* applies, though stronger measures might apply too. The *semantic resemblance* between the above two objects can be defined as :

semPro(O_1, O_2) = <LS$_1 \cup$ LS$_2$, _ , (D_1, D_2), _>
where D_1 = Domain(key(O_1))
and D_2 = Domain(key(O_2))
and   LS$_i$ = The local schemas to which objects O_i belong to (i = 1,2).
and   role-of(key(O_1), LS$_1$) = role-of(key(O_2), LS$_2$) = IDENTIFIER

7.2 Naming Conflicts

Semantically similar 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* of each other. Mappings between synonyms can often be established. In such cases objects O_1 and O_2 having this kind of a conflict can be considered to be *semantically equivalent*.

The schema correspondence between these two objects can be given as :

$<$ EMPLOYEE, $\{A_1,\ldots,A_n\}$, WORKER, $\{B_1,\ldots,B_n\}$, Ψ $>$
where Ψ is a total 1-1 value mapping defined as,
$\Psi : A_1 \times \ldots \times A_n \leftrightarrow B_1 \times \ldots \times B_n$

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. Since homonyms are semantically dissimilar, there cannot be any context, in which there is an abstraction which maps one homonym to another. Thus two objects O_1 and O_2 having this conflict can be considered to be *semantically incompatible*.

Note that the above conflicts are different from the *Naming Conflicts* discussed in Section 4.1 of this paper. The conflicts discussed in Section 4.1 arise due to differences in the naming of *attributes* whereas, conflicts in this section arise due to differences in the naming of *entities*.

7.3 Union Compatibility Conflicts

Descriptors of semantically similar entities might not be union compatible with each other. Two entities are union incompatible when the set of attributes are semantically unrelated in such a way that a one-one mapping is not possible between the two sets of attributes.

Example : Consider two entities STUDENT1 and STUDENT2 in different databases as follows :

```
            STUDENT1(Id#, Name, Grade)
            STUDENT2(Id#, Name, Address)
are two entities that are union incompatible.
```

Since mappings can be established between the objects on the basis of the common and identifying attributes, objects O_1 and O_2 can be considered to have a *semantically relationship*, i.e. their *semantic proximity* can be defined as follows:

semPro(O_1, O_2) = $<$ALL, Abstraction, (D_1, D_2), _$>$
where Abstraction is expressed in the following schema correspondence,

$<$ O_1, $\{A_1,\ldots,A_n\}$, O_2, $\{A_1,\ldots,A_n\}$, Ψ $>$

where $\{A_1,\ldots,A_n\}$ are the set of common and identifying attributes of O_1 and O_2 and Ψ is a partial many-one value mapping defined as,
$\Psi : O_1.A_1 \times \ldots \times O_2.A_n \leftrightarrow O_2.A_1 \times \ldots \times O_2.A_n$

7.4 Schema Isomorphism Conflicts

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

Example : Consider two entities INSTR1 and INSTR2 defined in different databases as follows :

```
            INSTR1(SS#, HomePhone, OffPhone)
            INSTR2(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 4.4 of this paper, as the Phone number of INSTR1 can be considered to be represented in a more precise manner than the Phone number of INSTR2. However, the conflicts discussed in section 4.4 are due to differences in the domains of the attributes representing the same information and hence are *attribute level conflicts*. Whereas, conflicts in this sections are *entity level conflicts* because they arise due to differences in the way the entities INSTR1 and INSTR2 are defined in the two databases.

Since mappings can be established between the objects on the basis of the common and identifying attributes, objects O_1 and O_2 can be considered to have a *semantic relationship*, i.e. their *semantic proximity* can be defined as follows:

semPro(INSTR1, INSTR2) = <ALL, Abstraction, (D_1, D_2), _>
where Abstraction is represented in the following schema correspondences,

$<$ INSTR1, $\{$SS#, HPhone$\}$, INSTR2, $\{$SS#, Phone$\}$, Ψ_1 $>$
where Ψ_1 could be a partial many-one value mapping as,
Ψ_1 : INSTR1.SS# $\times$ INSTR1.HPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone

$<$ INSTR1, $\{$SS#, OPhone$\}$, INSTR2, $\{$SS#, Phone$\}$, Ψ_2 $>$
where Ψ_2 could be a partial many-one value mapping as,
Ψ_2 : INSTR1.SS# $\times$ INSTR1.OPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone

7.5 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 conflicts discussed before.

There is 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 : Consider two entities STUDENT and GRAD-STUDENT define in different databases as follows :

```
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".
```

It should be noted that 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 4.5 which is an *attribute level conflict*. A potential resolution of the *entity level conflict* discussed in this section is based on the default value aspect of the *attribute level conflict* of section 4.5.

In this case, a mapping is possible between the objects, only after the value of the missing data item has been deduced. Hence, the process of deduction itself may be viewed as a mapping process. It is always possible to deduce a mapping *wrt* a context. Hence any two objects O_1 and O_2 having this kind of a conflict can be considered *semantically relevant*.

In the above example, before we are able to map the domains of the Type attributes in the two databases, we might have to use the generalization abstraction as,
$$\text{Student} = \text{Generalize(GRAD-STUDENT)},$$
and then we can introduce a partial 1-1 value mapping between the default values of the missing attribute(s).

semPro(STUDENT, GRAD-STUDENT) = $<$SAME, Abstraction, (D_1, D_2), _>
where Abstraction is expressed in the following schema correspondences :

$<$ STUDENT, {SS#, Name, Type}, GRAD-STUDENT, {SS#, Name}, Ψ $>$
where Ψ is a total many-one mapping defined as,
Ψ : STUDENT.SS# $\times$ STUDENT.Type $\times$ {"Grad", "UG"}
$\rightarrow$ GRAD-STUDENT.SS# $\times$ GRAD-STUDENT.Name
and Context = SAME = dependent on STUDENT.Type (*wrt* which the mapping has been deduced
and the schema correspondence defined above is associated with and depends on the above Context.
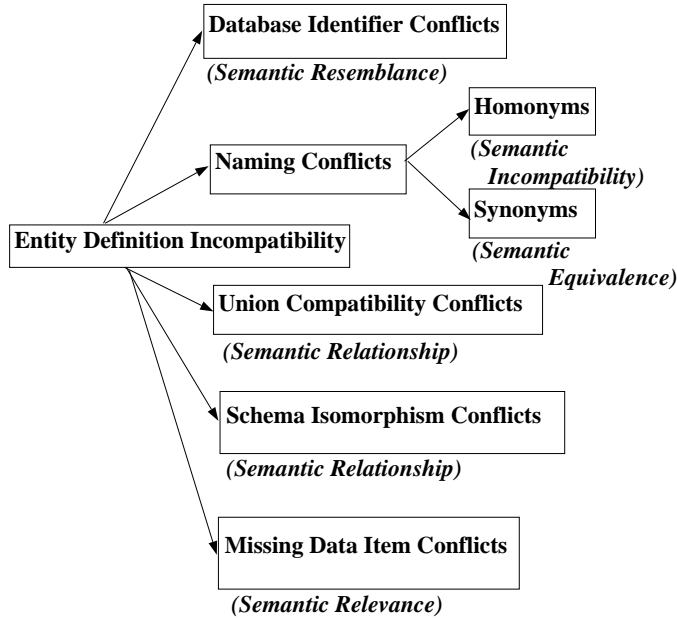
Figure 7: Entity Definition Incompatibilities and the *likely* types of semantic proximities

8 Data Value Incompatibility Problem

This class of conflicts covers those incompatibilities that arise due to the values of the data present in different databases [BOT86], and is different from the default value conflicts and attribute integrity constraint conflicts described in Section 5. The latter type of conflict is due to the definitions of the values of the attribute domains, whereas 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 :

```
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's values (database states) are inconsistent.
```

Below, we give a more detailed classification of the data value inconsistencies which can arise based on whether the cause of inconsistency is known and the extent and duration of the inconsistency. Also in the semantic classification of two objects having this class of conflicts, the state component of the semantic proximity descriptor plays an important

28

role because the conflicts here are in the extensions and not the schemas of the two objects. For each enumerated conflict, we specify the schema correspondence which in turn is associated with and a part of the definition of the semantic proximity between the objects and their extensions.

8.1 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. Here the cause of the inconsistency can be identified and the more reliable database can be used to resolve the inconsistency (e.g., overrule the less reliable database).

When the cause of inconsistency between objects is known ahead of time, it is possible to establish a mapping between objects having inconsistent values. However, the mappings might be between the (Domain, State) of the two objects. Hence, they may be considered to be *state semantically equivalent*, i.e., their semantic proximity can be defined as follows :

semPro(O_1, O_2) = <ALL, Abstraction, (D_1, D_2), (S_1, S_2)>
where Abstraction is expressed in the following schema correspondence,

< O_1, _, O_2, _, Ψ >
where Ψ is a total 1-1 value mapping given as,
Ψ : (D_1, S_1) $\leftrightarrow$ (D_2, S_2).

8.2 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*[6]. One of the databases which has conflicting values, might have obsolete information. This means that the information stored in the databases is time dependent. It is also possible that the change in information in one database has not yet propagated to the other databases.

In this case, since the inconsistency is only of a temporary nature, the objects may be said to be *eventually semantically equivalent*. In this case the semantic classification between two objects O_1 and O_2 depends on their states as well as time. Here we model the state of an object as a function of time. Thus the *semantic proximity* can be defined as follows :

semPro(O_1, O_2)=<ALL, Abstraction, (D_1, D_2), (S_1, S_2)>
where Abstraction is expressed in the following schema correspondence,

< O_1, _, O_2, _, Ψ >

---

[6]Additional information on weaker criteria for consistency can be found in the literature on transaction models (e.g., see [SRK92]).
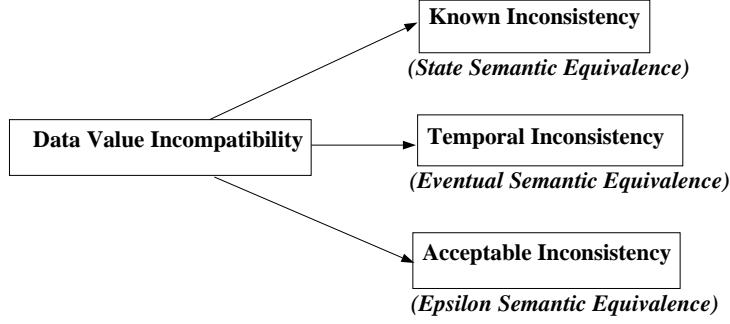
Figure 8: Data value incompatibilities and the *likely* types of semantic proximities

where Ψ is a total 1-1 value mapping defined by,
$\Psi : S_1(t + \Delta t) \leftrightarrow S_2(t)$ i.e. $S_2(t) = \Psi(S_1(t)) = S_1(t + \Delta t)$

8.3   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.

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.
```

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 towns in New Jersey), then again the inconsistency may
be ignored.
```

In this case, since the inconsistency between two objects O_1 and O_2 is considered to be acceptable, the two objects may be considered to be *epsilon semantically equivalent*. Thus, the *semantic proximity* can be defined as follows :

semPro(O_1, O_2)=<ALL, Abstraction, (D_1, D_2), (S_1, S_2)>
Abstraction is expressed in the following schema correspondence,

$< O_1,$ _, $O_2,$ _, $\Psi >$
where Ψ is a total 1-1 value mapping defined by,
$\Psi : \text{perturb}(S_1, \epsilon) \leftrightarrow S_2$ i.e.
$S_2 = \Psi(S_1) = \text{perturb}(S_1, \epsilon)$ where ϵ is the discrepancy in the state of the two objects.

9 Abstraction Level Incompatibility Problem

This class of conflicts was first discussed in [DH84] in the context of the functional model. These incompatibilities arise when two semantically similar entities are represented at differing levels of abstraction. Differences in abstraction can arise due to the different levels of generality at which an entity is represented in the database. They can also arise due to aggregation used both at the entity as well as the attribute level. For each enumerated conflict, we specify the schema correspondence which in turn is associated with and a part of the definition of the semantic proximity between the entities.

9.1 Generalization Conflicts

These conflicts arise when two entities are represented at different levels of generalization in two different databases. Also, there might be a natural inclusion relationship induced between the two entities.

Example : Consider the entities "Graduate Students" which may be represented in two different databases as follows :

```
            STUDENT(Id#, Name, Major)
            GRAD-STUDENT(Id#, Name, Major, Advisor)
Thus we have the same entity set being defined at a more general
level in the first database.
```

In this case there is an inclusion relationship between two conflicting objects and hence, they may be considered to have a *semantic relationship*.

semPro(STUDENT, GRAD-STUDENT) = <ALL, Abstraction, (D_1, D_2), _>
where Abstraction is expressed in the following schema correspondence :

< STUDENT, {Id#}, GRAD-STUDENT, {Id#}, Ψ >
where Ψ is a total many-one mapping defined as,
Ψ : STUDENT.ID# $\rightarrow$ GRAD-STUDENT.Id#

9.2 Aggregation Conflicts

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

Example : Consider the aggregation SET-OF which is used to define a concept in the first database and the set of entities in another database as follows :

```
            CONVOY(Id#, AvgWeight, Location)
            SHIP(Id#, Weight, Location, Captain)
Thus, CONVOY in the first database is a SET-OF SHIPs
```
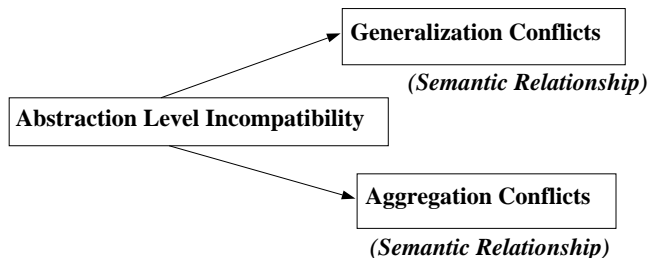
Figure 9: Abstraction level incompatibilities and the *likely* types of semantic proximities

Also, CONVOY.AvgWeight is the average (aggregate function) ship weight of the convoy.

In this case there is a mapping in one direction only, i.e. an element of a set is mapped to the set itself. In the other direction, the mapping is not precise. When the SHIP entity is known, one can identify the CONVOY entity it belongs to, but not vice versa. Hence two objects might be considered to have a *semantic relationship*. Thus, the *semantic proximity* can be defined as follows :

semPro(SHIP, CONVOY) = $<$ALL, Abstraction, (D_1, D_2), _$>$
where Abstraction is expressed in the schema correspondence,

$<$ SHIP, {Id#, Weight}, CONVOY, {Id#, AvgWeight}, Ψ $>$
where Ψ is a partial many-one mapping defined as,
Ψ : SHIP.Id#, $\times$ Weight $\rightarrow$ CONVOY.Id# $\times$ AvgWeight

10   Schematic Discrepancies Problem

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 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), ... }
........
```

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. Note 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).

10.1 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*.

Since this conflict is dependent on the database state, the fourth component of the 4-tuple describing the semantic proximity plays an important role. Also the mappings here are established between set of attributes ($\{O_i\}$) and values in the extension of the other attribute (O_2). Thus the two objects may be considered to be *meta semantically equivalent* and their *semantic proximity* can be defined as follows :

semPro($\{O_i\}$, O_2) = <ALL, Abstraction, (D_1, D_2), (S_1, S_2)>
where Abstraction is expressed in the schema correspondence,

< $\{O_i\}$, ϕ^7, O_2, $\{A\}$, Ψ >
where A $\in$ attr(O_2) such that it's values match the attributes O_i
and Ψ is a total 1-1 mapping defined as,
Ψ : $\{O_i\}$ $\leftrightarrow$ $S_2.A$

10.2 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 classified as a subclass of the entity definition incompatibility problem. Referring to the example described in the beginning of this section, the attributes *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 to be *semantically equivalent* 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

---

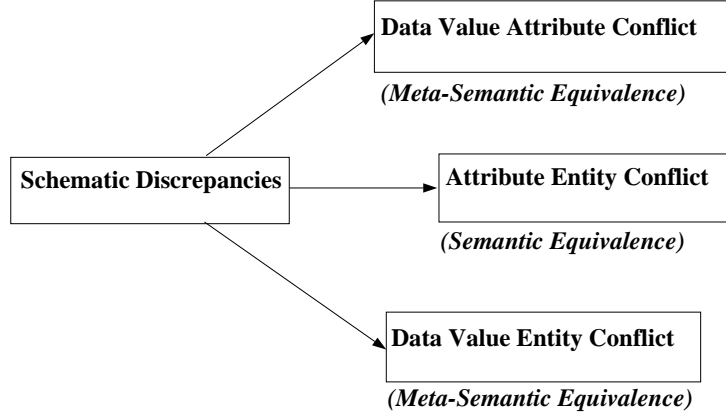[7]Since O_i's model information at the attribute level.

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

(property) and O_2 is an entity (object class). Thus the *semantic proximity* can be defined as follows :

semPro(O_1, O_2) = <ALL, Abstraction, (D_1, D_2), _>
where Abstraction is expressed in the schema correspondence,

< O_1, ϕ, O_2, {Identifier(O_2)}, Ψ >
where Ψ is a total 1-1 value mapping defined as,
Ψ : O_1 $\leftrightarrow$ Identifier(O_2)

10.3   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*.

Since this conflict is dependent on the database state, the state component of semantic proximity plays an important role. Also the mappings here are established between set of entities ({O_i}) and values in the extension of an attribute (O_2). Thus the two objects may be considered to be *meta semantically equivalent* and their *semantic proximity* can be defined as follows :

semPro({O_i}, O_2) = < ALL, Abstraction, (D_1, D_2), (S_1, S_2)>
where Abstraction is expressed in the schema correspondence,

< {O_i}, _, O_2, {A}, Ψ >
where A $\in$ attr(O_2) such that it's values match the entities O_i
and Ψ is a total 1-1 mapping defined as,
Ψ : {O_i} $\leftrightarrow$ S_2.A

11   Conclusion

An essential prerequisite to achieving interoperability in a multidatabase environment is to be able to identify semantically similar data in different database systems. In this paper we have defined the concept of *semantic proximity*, using which we represent the degrees of similarities between the objects [SK92]. We use the *context* in which these objects are being compared to capture the real world semantics of the relationships. The capability to combine semantically similar data from different databases is required to process queries spanning multiple databases. We have defined a comprehensive *structural taxonomy* of the various schematic (structural, representational) conflicts between objects having some semantic similarity. We have defined the concept of *schema correspondences*, using which we represent the structural similarities between objects having schematic conflicts. We use the mechanism of *domain value mappings* to capture the structural similarities.

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 and as component(s) of the semantic proximity among objects. This association between the schema correspondences and semantic proximity enables us to determine measures of semantic similarity viz. *equivalence, relationship, relevance, resemblance and incompatibility*. Based on these similarity measures, we develop a *semantic taxonomy* which emphasizes the similarities between objects. We show the relationship of this taxonomy to the *structural taxonomy* which emphasizes schematic (representational, structural) differences among the objects [SK92]. The association between the schema correspondences and semantic proximity helps represent the possible semantic similarities between two objects having these conflicts.

We demonstrate how the semantic proximity between two objects can serve as a basis for the assignment of fuzzy strengths to specify the degree of semantic similarity between them. We define fuzzy strengths as a function of the semantic proximity and establish criteria for a consistent assignment of fuzzy values based on the measures of semantic similarity mentioned above. We also discuss the various types of inconsistencies possible between semantically similar data in different databases and model them using semantic proximity. Thus we establish *uncertainty* and *inconsistency* as aspects of *semantics*.

Work is in progress on the following research issues :

- We have demonstrated the reconciliation of the schematic and semantic perspectives. The development of resolution techniques for schematic conflicts [KCGS93, HM93] is complementary to the discussion in this paper. We need to incorporate semantic similarities between objects represented by the semantic proximity in the conflict resolution techniques defined above.

- Additional work is needed to further clarify the nature and structure of the context in which two objects are defined. Our work in this area is reported in [SK93]. Experimentation is required with different representations of context and methodologies devised for their efficient management.

- In this paper we approximate the relationship between strucuture and semantics

by viewing the schema correspondences as a *projection* of semantic proximity *wrt* context. We need to define this projection operation.

- We have expressed the fuzzy strengths as functions of semantic proximity. An investigation into uncertainty functions depending on semantic proximity in general and context in particular is also required.

- We are investigating the application of the dual semantic and schematic perspectives to information discovery and identification [KS93].

Acknowledgements

References

[BGMP90]   D. Barbara, H. Garcia-Molina, and D. Porter. A Probabilistic Relational Model. *Lecture Notes in Computer Science : Advances in Database Technology EDBT '90*, #416, 1990.

[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.

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

[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.

[DeM89]   L. DeMichiel. Resolving Database Incompatibility : An approach to performing Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4), 1989.

[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*. Ben-
           jamin/Cummins, 1989.

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

[GB91]   D. Gangopadhyay and T. Barsalou. On the Semantic Equivalence of Hetero-
           geneous Representations in Multimodel Multidatabase Systems. *SIGMOD
           Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed.,
           20(4), December 1991.

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

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

[HM93]   J. Hammer and D. McLeod. An approach to resolving Semantic Hetero-
           geneity in a Federation of Autonomous, Heterogeneous, Database Systems.
           International Journal of Intelligent and Cooperative Information Systems.,
           March 1993.

[KCGS93]   W. Kim, I. Choi, S. Gala, and M. Scheevel. On resolving Schematic Hetero-
           geneity in Multidatabase Systems. *Distributed and Parallel Databases, An
           International Journal*, 1(3), July 1993.

[Ken91]   W. Kent. The breakdown of the Information Model in Multidatabase Sys-
           tems. *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 Interop-
           erability of Databases with Schematic Discrepancies. In *Proceedings of 1991
           ACM SIGMOD*, May 1991.

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

[KS93]   V. Kashyap and A. Sheth. Context based arbitration for information dis-
           covery and multidatabase interoperability. In preparation ...., 1993.

[LA86]   W. Litwin and A. Abdellatif. Multidatabase Interoperability. *IEEE Com-
           puter*, 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.

[ME84]   M. Mannino and W. Effelsberg. Matching techniques in Global Schema Design. In *Proceedings of the 1st IEEE Conference on Data Engineering*, April 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.

[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.

[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.

[SK93]   A. Sheth and V. Kashyap. Context-bound Semantics (a Multidatabase perspective). To be submitted for journal publication, 1993.

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

[SM91]   M. Siegel and S. Madnick. A Metadata Approach to resolving Semantic Conflicts. In *Proceedings of the 17th VLDB*, September 1991.

[SM92]   A. Sheth and H. Marcus. Schema Analysis and Integration: Methodology, Techniques and Prototype Toolkit. Technical Report TM-STS-019981/1, Bellcore, March 1992.

[SPD92]   S. Spaccapietra, C. Parent, and Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *The VLDB Journal*, July 1992.

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

[TCY93]   F. Tseng, A. Chen, and W. Yang. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases, An International Journal*, 1(3), July 1993.

[UW91]   S. Urban and J. Wu. Resolving Semantic Heterogeneity through the explicit representation of Data Model Semantics. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[VH91]   V. Ventrone and S. Heiler. Semantic Heterogeneity as a result of Domain Evolution. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[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.

[Zad78]   L. Zadeh. Fuzzy Sets as a basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1(1), 1978.