Journal of Intelligent Information Systems, 3, 151–183 (1994)
© 1994 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Coordinating Context Building in Heterogeneous Information Systems

ARIS M. OUKSEL[1] AND CHANNAH F. NAIMAN M.ARIS.OUKSEL@UICVM.UIC.EDU
The University of Illinois at Chicago Dept. of Information and Decision Sciences (M/C 294), PoB 802451, Chicago, IL 60680-2451

Abstract. We present an architecture to coordinate the construction of the context within which meaningful information between heterogeneous information systems can be exchanged. We call this coordinator SCOPES (Semantic Coordinator Over Parallel Exploration Spaces). A classification of semantic conflicts we proposed elsewhere is used to build and refine the context, by discovering the semantic mapping rules (inter-schema correspondence assertions) between corresponding elements of the communicating systems. A truth maintenance system is used to manage the multiple intermediate contexts. It provides a mechanism to infer or retract assertions on the basis of the knowledge acquired during the reconciliation process. This nonmonotonic technique is used in conjunction with the Dempster-Shafer theory of belief functions to model the likelihood of alternative contexts. Finally, we propose an algorithm which illustrates how the various components of the architecture interact with one another in order to build context.

Keywords: Heterogeneous information systems, semantic conflicts, context, plausible assertions, truth maintenance systems, coordination.

When you have eliminated the impossible, whatever remains, however improbable, must be the truth
— Sherlock Holmes (A. Conan Doyle)

1. Introduction

1.1. Communication within Context

Distribution and Coordination. Organizations are often geographically distributed, functionally diverse and architecturally fluid. Increasingly, businesses and people require and expect access to diverse sources of information (Elmagarmid and Pu, 1990) to develop new applications and to implement business processes. Communication among these cooperating systems is, however, complicated by the complexity of both the information being exchanged and the technology that provides the vehicle for that information. Underscoring this complexity is the heterogeneity of the communicating information systems. In this paper, we examine the differences that can exist among information systems, and we focus on the coordination that is needed to reason about information that is exchanged between heterogeneous systems.

Semantic Communication. Information systems can be heterogeneous on several levels, ranging from their use of different data models to slight formatting differences of similar instances. These heterogeneities are reflected in the structure of the database representation, and also in the semantics of the data being represented (Sheth and Larson, 1990). The exchange of data must be understood within the semantic interpretation intended by the different representations, so that the semantic differences of those data can be recognized and, if possible, reconciled. If reconciliation is not achieved, intelligent cooperation cannot occur.

Integration of Heterogeneous Systems. All of the informational needs that exist in the heterogeneous environment presuppose heterogeneous databases, and require some kind of integration or interoperability among them. Interoperability between heterogeneous systems requires two distinct phases. The *discovery* phase includes the examination of the component schemas, and reconciles their semantic differences. The result of the discovery phase is a set of inter-schema correspondence assertions, or less formally, any *mapping of the semantic concepts* between the component databases. The *integration phase* includes the conversions and translations that permit navigation of another schema. The result of the integration phase is a *structural schematic mapping* between the component databases. Techniques applied in the discovery phase focus on identifying equivalent or corresponding objects or attributes across databases. For example, Larson, Navathe and Elmrasri (Larson, et al., 1989) address the problem of attribute equivalence; Siegel and Madnick (Siegel and Madnick, 1991) enrich the attribute domain semantics with metadata, thereby providing a better semantic mapping of corresponding attributes; and Wang and Madnick (WaMade89) rely on the availability of instances of a particular attribute being examined as well as instances of related attributes in order to resolve naming and format conflicts of the values. These and similar techniques have in common that they map similar semantic concepts across databases.

In all cases, a semantic mapping must precede the structural schema integration. Integration itself can be achieved using a global schema (Batini, et al., 1986), or the definition of join fields (Chatterjee and Segev, 1993), indices (Kent, 1979), routing tables (Giladi and Shoval, 1993) or other information to facilitate the schematic mapping. Schema integration often requires mapping to a canonical data model (Bertino, 1990; Ram and Barkmeyer, 1991; Urban, 1991), or mapping the local query to a global Data Manipulation Language (Chomicki and Litwin, 1992; Qian, 1993). Most of the approaches proposed for providing automatic or semi-automatic schema integration focus on the integration phase, with the discovery phase acknowledged as having been completed prior to the application of the technique.

The SCOPES Coordinator. While most semantic reconciliation and schema integration techniques operate under assumptions of prior semantic knowledge, each technique addresses a slightly different semantic conflict, and each may require a slightly different set of semantic assumptions as preconditions to its application. The prior semantic knowledge required by a specific technique is the *context* within which it can be applied. We argue that the more flexible interoperability that will be required of next-generation information systems mandates a process of semantic reconciliation that does not depend upon complete prior semantic knowledge of the communicating systems. We also suggest that semantic reconciliation should not be limited to a single technique; rather, at any point in the reconciliation process, the most appropriate available technique should be applied to a given semantic conflict, within the context of the semantic knowledge that has been acquired to that point in time. Clearly, our approach requires a coordination mechanism to recognize a semantic conflict, to assign a technique to that conflict, and to place into context the semantic knowledge that is acquired by applying the reconciliation techniques. In contrast to the static acquisition of a comprehensive context, as is required for static global integration, we propose an architecture to support the incremental building of context in order to achieve partial, dynamic integration. We call this architecture SCOPES: Semantic Coordinator Over Parallel Exploration Spaces. The purpose of SCOPES is to exploit the available reconciliation techniques, and by coordinating their application, to increase the automation of the semantic reconciliation process. A cardinal characteristic of SCOPES is the incremental acquisition of knowledge, in order to dynamically build a context for goal-oriented reconciliation. SCOPES supports interconnectivity, while addressing the problems of heterogeneity. In this sense, SCOPES is an enabling technology for the cooperative processing applications that require an open systems (Hewitt, 1985) environment.

1.2. Organization of the Paper

Semantic reconciliation has conventionally been achieved by examining the component schemas, and statically building either a global schema or, alternatively, join tables or semantic routing tables. In Section 2, we discuss the motivation for a dynamic methodology for the semantic reconciliation process. The dynamic process would involve a more explicit recognition of the information needed to build context and of how that context would affect the interpretation of semantic conflicts. We discuss the definition of context, as well as its interpretation, organization and maintenance, in Section 3. Having motivated the need for dynamic reconciliation and described a mechanism by which the needed context can be managed in Section 4. Section 5 introduces the SCOPES architecture. We outline the components of SCOPES, which are required for the dynamic reconciliation process. We propose in Section 6 an algorithm which demonstrates the coordination of the SCOPES components in achieving dynamic reconciliation. We present our conclusions, as well as a discussion on future work, in Section 7.

2. Motivation for Dynamic Integration

2.1. Problems of Building a Global Schema.

Building a global schema involves mapping the schemas of the component databases to a single schematic representation. A series of steps has been proposed by (Sheth and Larson, 1990) and by (Batini, et al., 1986) to govern the manipulation of the schematic and semantic information that is needed in order to recognize and reconcile the semantic conflicts among the participating schemas. Once object identification and format standardization are achieved, mapping to the global view is still not trivial. Furthermore, management of the global database entails determining policy issues of participation and control as well as the complicated update problems of integrity and concurrency. These update problems exist not only for a global schema, but for other structures that support interoperability, such as join tables or routing tables, if those structures require a static "snapshot" of the participating schemas and their metadata. Underlying these issues is the more fundamental concern that the participating databases may not find it desirable to conform to the standards of access set by the federation; alternatively, it may not always be possible to create a global schema with all relevant systems.

2.2. *Static vs. Dynamic Integration*

Many of the features that will be important to next-generation information systems are not characteristic of static integration, but can more easily be incorporated in systems providing dynamic integration. These features include *autonomy*, the ability of the communicating systems to determine the level of participation and extent of data access in any interaction; *flexibility*, the ability of an integrator to recognize a semantic conflict at any level of heterogeneity, and to tailor the reconciliation process according to the knowledge and the techniques that become available; *extensibility*, the ability to connect, semantically, with new information sources (Wiederhold, 1992); *semantic communication independence*, the ability of each system to choose the set of tools to maintain, consonant with the level and extent of communication and coordination with other databases, that it wishes to support; *transparency* in the coordination of the knowledge acquisition process, and in the dialogue between systems; and *scalability*, the ability to use the same reconcil iation methodology on an increasingly large number of remote systems;

The above characteristics have been widely recognized as desirable goals of interoperability. For instance, Sheth and Larson (Sheth and Larson, 1990) stress the importance of autonomy and transparency, while Rosenthal and Siegel (Rosenthal and Siegel, 1991) consider flexible usage, integrator customization and extension to be "crucial". They also discuss the importance of incremental development of the information about another system, and they emphasize "method extensibility", which is similar to what we call semantic communication independence. The importance of a dynamic discovery process is also emphasized in (Fang, et al., 1992), and the feasibility of such an autonomous, active and self-modifying federated database system has been shown in (Litwin, 1989). Litwin, Mark and Roussoupoulos (Litwin, et al., 1990) review in detail the problems of interoperability among autonomous databases.

We have designed SCOPES to include the important characteristics of dynamic integration. The design objectives of SCOPES are described later along with the architecture. An important **caveat**: we do not claim that SCOPES is automatic in all its aspects. Our objective is to understand the interplay between the various sources of knowledge and the underlying coordination mechanisms to build context. We endeavor to make explicit that knowledge which a human integrator uses implicitly to resolve a semantic conflict. We propose the SCOPES system to provide a framework into which automatic techniques can be integrated as they become available.

Database: *A database of a cold-cereal manufacturer*

Data Model: *Relational*

Schema(partial):

COMPETITOR(NAME,LOCATION,MKT_SGMT,MKT_SHR,QTLY_INCOME)

Competitor: any tuple of the relation COMPETITOR, identified by the key NAME.
Performance: market share and quarterly gross income
Market Share: percentage of the cereal market of that market segment, controlled by the competitor, expressed as .99.
Quarterly Income: Gross income, as reported on the last quarterly statement, expressed in ten-thousands.
Europe: a defined market segment
Location: location of the headquarters of that COMPETITOR, such as "Kellogg's, is located in Battle Creek, Michigan.

Figure 1a. Local Database Schema

3. Building Context

3.1. Context

We describe context as the knowledge that is needed to reason about another system, for the purpose of answering a specific query. The exact knowledge may differ from query to query, or from system to system, depending on the semantic information obtained, or on the tools that are available to interpret that information. Furthermore, this context must be organized in a way that facilitates the reconciliation process: context must provide an easily understood representation of how much is known and consequently of what is still needed in order to answer the query. In the rest of this section, we illustrate with an example how different contexts can lend different interpretations to a schematic conflict; we briefly discuss our classification of semantic conflicts, which has been developed elsewhere (Naiman and Ouksel, 1994) using our example; we examine the processes of detection and integration that are specific to a system that dynamically builds context; and finally, we discuss a method of organizing and maintaining context built by such a system. We shall use the partial schemas given in Figures 1a., 1b., and 1c. to elaborate on the complex issues of context building.

Throughout, the query given below, posed against the local database, initiates the interaction with the other databases.

Query: "What is the performance of my competitors in Europe?"

Database: *This database maintains financial information of companies in the European Community*

Data Model: *Relational*

Schema(partial):

COMPANY(RANK,CO-NAME,LOCATION,INDUSTRY,SALES)

Figure 1b. Target Database #1

Database: *This database maintains product information for a consumer advocacy organization.*
Data Model: *Object-oriented*

Schema(partial):

```
PRODUCT
┌──────────────
| Name
| Producer-Manufacturer--------------PRODUCER-MNFCTR
| Market-share                   ┌──────────────
| Sales                          | Name
└────────────           |          Headquarters
                                 | ...

 FOOD PRODUCT(subclass of PRODUCT)
 ┌───────
 | ...
 | ...
 | ...
 └──────

  CEREAL(subclass of FOOD PRODUCT)
  ┌───────
  | ...
  | ...
  | ...
  └──────

   COLD-CEREAL(subclass of CEREAL)
   ┌────────────────
   | ...
   | Sugar-Levels
   | Sweetener
   | Health-Category
   └────────────────
```

Figure 1c. Target Database #2

3.2. Context-Dependent Interpretation

To illustrate context-dependent interpretation, we will focus on the mapping of **COMPETITOR.Name** to the remote databases. At first glance, the attributes **COMPETITOR.Name** and **PRODUCT.Name** appear to be synonyms, This conclusion must be re- examined in the context of the objects: since **COMPETITOR** and **PRODUCT** are unrelated, we would most likely conclude that the two attributes may in fact be homonyms, that is, that they are similar terms referring to different semantic concepts. A similar reasoning process can be applied to the attributes **COMPETITOR.Name** and **COMPANY.Co-Name**. However in this case, **COMPETITOR** and **COMPANY** are not entirely unrelated: in fact, the term "competitor" is a specialization of the term "company". The question to be addressed at this point involves the relevance of that specialization in this particular case: does the generalization/specialization relationship between "company" and "competitor" provide knowledge that can be used to interpret the relationship between the two objects for the purpose of this query? The query itself may provide the definitive context: if the query had specified the term "company" or "organization" or a similarly general term, then **COMPETITOR** and **COMPANY** may be interpreted as equivalent objects in that context. This query specifies "competitor". Thus in the context of the query, the object **COMPETITOR** may be related as a specialization of the object **COMPANY**. By implication, we may conclude that **COMPETITOR.Name** and **COMPANY.Co-Name** are synonyms for the corresponding object instances.

Even the query, however, must be understood within the context of the databases themselves. The nature of the information maintained in the databases may provide a context that corroborates or contradicts an otherwise reasonable conclusion. For example, if the databases maintain information about very different types of businesses, this may support the conclusion that **COMPETITOR** and **COMPANY** are semantically unrelated, and thus, in the context of the databases, Name and Co- Name may, after all, be homonyms. Clearly, context becomes a crucial issue in resolving the ambiguity that arises in interpreting the semantic conflicts in this case.

3.2.1. Organizing Context by Levels of Heterogeneity

In understanding and representing context, we take our cues from the special case where full schematic and semantic information of the external system is available to the local system. This is the case of the statically integrated global schema, as surveyed in (Batini, et al., 1986), which we have observed, achieves reconciliation by gathering semantic information at progressively more refined levels of schematic heterogeneity. The object level is considered to be a coarser or higher level of heterogeneity than the attribute level, which is coarser than the instance level. In addition to structural schematic levels of heterogeneity, there are *metadata* levels of heterogeneity. These include the differences that require knowledge describing the objects and attributes of the schema (commonly called *descriptive metadata*), knowledge of the semantics inherent, implicit and explicit in data models (see Brodie, 1984, for a discussion), and whatever general or domain-specific knowledge is available about the database itself. The *schematic level* is specific to the schematic elements. The *data model level* is more general–it is knowledge that pertains to the entire schematic structure, and the semantic relationships that can be inferred from that structure. (Semantic enrichment or data model mapping tools such as those in (Castellanos, 1993; Ioannidis and Livny, 1989; Meng, 1992) discuss the semantics that may be understood from the structure of a data model.The *database level* includes information about the domain of the organization, as well as the nature of the information stored in the database, which is data model independent. In our example, the database-level metadata of the Target database #2 specifies that this database maintains information about products in the European Community for a consumer advocacy organization. The context of the query drives the reconciliation process: the query terms constitute those elements that require mapping to another database.

In the above example of **COMPETITOR.Name** and **COMPANY.Co-Name**, for instance, the *Inter Schema Correspondence Assertion* (ISCA) that the attributes are corresponding attributes *triggers* an attempt to place those attributes in the context of the respective objects that they describe. This may result in the inference of an ISCA that a possible relationship exists between **COMPETITOR** and **COMPANY**, which merits further examination by the reconciliation techniques. We call such a trigger, from a lower level of heterogeneity to a higher level, *upward propagation* through the levels of heterogeneity. In our example, if there is enough evidence to refute the synonymy of **COMPETITOR** and **COMPANY** this would trigger a *downward propagation* through the levels of heterogeneity to ultimately reclassify the relationship of the two attributes as, for instance, homonyms. Upward or downward propagation is the navigation through the different levels of heterogeneity in order to acquire knowledge as well as to assert correspondences based on that knowledge. A single reconciliation technique can trigger both upward and downward propagation. For example, the entity identification technique of Lim, et al. (Lim, et al., 1993) triggers upward propagation by requiring environmental knowledge (*database- level metadata*) in order to extend the global key; the technique also triggers downward propagation since it requires examination of the attributes (*attribute-level structural schematic knowledge*) in order to evaluate the similarity between the entities.

Our approach of organizing semantic metadata into schematic, database and data model metadata turns the focus on to the discovery of the metadata that can be useful in resolving semantic conflict. This is in contrast to many techniques, which focus on the integration phase, with the discovery phase acknowledged as having been completed prior to the application of the technique. For example, in (Spaccapietra, et al., 1992), the technique provides a schematic mapping, given that the inter-schema correspondence assertions have been previously specified. Other approaches, such as (Chatterjee and Segev, 1993; Sciore, et al., 1993) focus on the representation of this semantic metadata; the representation, however, does not assist in the discovery process. For instance, (Sciore, et al., 1993) explicitly represents the relevant metadata as "semantic values", which are properties that are supposed to describe the context of the attribute. However, that context is statically defined, and is simply a description of the same attribute, at a lower level of granularity. The same semantic conflicts that can arise between attributes can arise between these more refined properties of attributes. In (Chatterjee and Segev, 1993), these semantic properties are represented as components of a vector. They attempt to explicitly represent the metadata that implicitly provides a contextual interpretation for an attribute. Once again, the problem lies not as much in the representation of this semantic information as in its discovery: it is specifically this implicit semantic knowledge that is *least likely* to be explicitly recognized. Our approach would explicitly explore *database metadata* in an effort to discover untapped "environmental" information that could clarify the context of this attribute. When full semantic knowledge is available, conflicts at the higher levels are resolved before the next level is addressed. Thus, the semantic knowledge of another system, organized by levels of heterogeneity, provides a context for interpreting the view of that system. *In a dynamic system, when full semantic knowledge is not available, knowledge is not gathered in the same order as that required by the static resolution of the semantic conflicts. Knowledge is acquired in a piecemeal fashion through upward and downward propagation. When a piece of knowledge is discovered, it may be relevant to a semantic conflict which cannot be resolved until other conflicts have been resolved. Similarly, an export schema, or any available semantic knowledge, can be exploited to further the reconciliation process. Thus, the organization and the maintenance of the acquired knowledge becomes crucial to the precedence relationships of the dynamic reconciliation process.*

3.3. Classification and Representation of Semantic Conflicts

We have developed a classification of semantic conflicts, which is more fully described in (Naiman and Ouksel, 1994). It is a fundamental block of dynamic context building, and thus it is briefly summarized below.

We classify semantic conflicts along the three dimensions of naming, abstraction and level of heterogeneity. The semantic relationship between two elements of different databases is represented as an inter-schema correspondence assertion, with the following general form:

Assert $[x,y]$ (naming,abstraction,heterogeneity)

where x refers to an element in the local database schema and y refers to an element in the target database schema. A unique feature of this classification is that it combines the inherent dimensions of semantic conflicts with a structural description that provides for operational integration. The first two dimensions (naming and abstraction) include what we maintain are the fundamental relationships between semantic concepts. The third (level of heterogeneity) is needed in order to place the semantic relationship in the appropriate schematic context.

(i) **Naming** *Naming Conflicts* refer to the relationship of the object, attribute or instance names. These conflicts include *synonyms* and *homonyms*. Alternatively, a conflict can be classified as *unrelated* if the corresponding elements cannot be categorized as either synonyms or homonyms. The naming relationship is commutative, and can be understood as, for instance, "x is a synonym of y", or equivalently, "y is a synonym of x".

(ii) Abstraction A conflict can involve objects that refer to the same *class* of objects, objects that represent similar semantic concepts at different levels of abstraction (*generalization/specialization*), an object that maps to a group of objects in another database (*aggregation/part-of*, or an incompatibility that occurs when one object can be mapped to another through a *computed or derived function* (Fang, et al., 1992). The abstraction relationship is directed from x to y, for example, "x is a generalization of y".

(iii) Levels of Heterogeneity. Naming and abstraction conflicts can exist at the object, attribute and instance levels of the schema. Two values are required for this dimension, one for each element, as represented in its respective schema.

Referring to the example of **COMPANY.Co-Name** and **COMPETITOR.Name**, this conflict could be classified as:

Assert[COMPANY.Co-Name,COMPETITOR.Name]
(synonyms,generalization,(attribute,attribute))

Assertions that have been inferred through upwards propagation require reconciliation to fill the slots of the classification. For instance, the previous assertion implies

Assert[COMPANY,COMPETITOR](?,?,(object,object))

Complex conflicts can be expressed in disjunctive normal form: For instance, the following assertions express the generalization relationship between **COMPANY** and **COMPETITOR** with the specialization defined on the attribute **Industry**.

Assert[COMPANY,COMPETITOR] (synonyms,generalization,(object,object))
AND **Assert**[COMPANY[i].Industry."Cereal",COMPETITOR[j]]
(synonyms,class,(instance,instance)))
OR **Assert** [COMPANY,COMPETITOR] (synonyms,class,(object,object))

The first assertion asserts a generalization relationship between **COMPANY** and **COMPETITOR**. The second assertion asserts a correspondence between an instance i of **COMPETITOR** and an instance j of **COMPANY**, constrained by the value **"Cereal"** for the attribute **COMPANY.Industry**. Semantic conflicts classified as ISCAs constitute the semantic understanding of another system's schema (its schematic context) at any given point in time. It is this context which supports automatic detection of conflict in SCOPES.

Note that the number of possible cases for classification for each case is the cartesian product of {Naming} X {Abstraction}, which is twelve cases. (The level of heterogeneity is used as a bookkeeping mechanism in order to facilitate schema navigation.) But as we shall see later, not all of these cases are examined during reconciliation.

3.4. The Process of Conflict Detection

An automatic detection system must anticipate what knowledge sources may be germane to the problem, and must provide a mechanism for their systematic exploitation. In effect, an automatic detection system must attempt to explicitly formalize the implicit free association process of manual integration. The extent to which such a system can be successful lies in the state of the art of artificial intelligence and knowledge representation. While technology may never achieve the level of nuance and subtlety of human understanding and knowledge, rules and constraints, as well as exceptions to those rules can be recognized. The challenge then becomes to identify and formalize those rules, constraints and exceptions, and to incorporate as many relevant knowledge sources, both schematic and domain specific, as is feasible and manageable.

3.5. Query-Directed Elicitation of Knowledge

Our methodology to eliciting this explicit knowledge can be characterized as query-directed partial integration. Its purpose is to identify the knowledge that is needed in order to detect only those conflicts whose resolution are necessary to answer a specific query. In this subsection, we demonstrate the explicit knowledge that is needed in order to answer the query in our example. We list the ISCAs for the query terms, and for each ISCA, we briefly discuss the knowledge that is needed to achieve such a mapping. This set of ISCAs represents the context that needs to be dynamically built for this query. In the next section, we discuss the truth maintenance system that we propose in order to build and manage this dynamic context.

For purposes of clarity and precision, we rewrite the query in SQL form:

```
Select NAME, MKT-SHR, QTLY-INCOME
    from COMPETITOR
    where MKT-SGMT = “Europe”
```

This query, while straightforward, is not easily mapped to the target databases. It is, nevertheless, the type of information that might reasonably be requested from other systems. We examine the process of obtaining knowledge from each target database:

3.5.1. Mapping to Target Database #1

To build a context within which the local and the target database can cooperate to answer the query, the target database is first explored to find a match for the original query terms. This *initial exploration* step does not yield any result in the case of our example. Therefore, for context building to proceed, new terms related to those in the query are needed to resume the exploration. The new terms may be obtained from knowledge sources such as concept hierarchies (Yu, et al., 1991), thesauruses, linguistic support tools or information retrieval techniques such as lexical analysis. We refer to the process of enlarging the set of search terms as *Term Expansion* through Naming or Abstraction. In our example, because we assumed that the domain of discourse is the business environment, the generalization relationship between the terms "company" and "competitor" is recognized. The term "company" is added to the search terms. Since it exists in the target database, the object **COMPANY** becomes an *anchor*, or a point of reference from which the target schema can be explored. The knowledge of the generalization relationship between the terms "company" and "competitor" has implications for the relationship between the objects **COMPANY** and **COMPETITOR**. The possible correspondences between the objects can, however, only be inferred with the explicit knowledge of the concept of generalization, and its implications for the relationships between objects. In our example, the generalization relationship between the two terms implies either a generalization or a class relationship between the two objects. Both correspondences are tentatively asserted. The decision as to which, if any, of the correspondences is correct will be based on the collection of *corroborating* or *contradictory* evidence. For example, through *downward propagation*, the attributes and, ultimately, the instances of **COMPETITOR** and **COMPANY** are compared. *Reconciliation techniques* such as (Garcia-Solaco, et al., 1993) are invoked to identify the subset of **COMPANY** to which **COMPETITOR** should be mapped. The process of identifying the precise schematic elements in the target database to which **COMPETITOR** may be matched is called *object identification.* Object identification relies not only on comparison of schematic elements, but also on the examination of database metadata. For example, the reconciliation technique by (Sciore, et al., 1993) formally incorporates metadata into the schema, in order to enrich the attribute semantics. In our example, knowledge of the generalization abstraction, as well as the schematic knowledge learned through downward propagation, lead to a search for the attribute along which **COMPETITOR** can be mapped to **COMPANY**. Let us assume that database metadata, gathered through *upward propagation*, allows us to infer that a competitor is a company within the same "industry". The attribute **COMPANY.Industry** matches the term "industry" in the metadata description. The understanding that **COMPETITOR** is a specialization of **COMPANY**, partitioned along the attribute **COMPANY.Industry** is *corroborating evidence* of the generalization relationship between the two objects, and is *contradictory evidence* of the class relationship.

We shall examine below how the knowledge gathered by reconciliation techniques at the different levels of heterogeneity, or by exploitation of database metadata, provides the constraints to confirm correct assertions and to refute incorrect assertions. Assertions are confirmed either permanently if the evidence is irrefutable or temporarily if the supporting evidence is incomplete. Refuted assertions, are either rejected from permanently, if the contradictory evidence is complete, or temporarily if it is partial.

In our example, the correct assertion which must be asserted is the generalization assertion between the two objects **COMPETITOR** and **COMPANY**.

Assert[COMPETITOR,COMPANY] (synonym,specialization,(object,object))

Let us now detail the process that leads to the corroboration of this assertion. In other words, what is the evidence that supports the confirmation of this assertion, and how is this evidence gathered?

Attribute-Level Conflict Detection. The application of the object identification reconciliation techniques at the attribute level would result in the collection of the other attributes in the corresponding objects. Among these attributes are **COMPANY.Co-Name**, **COMPANY.Location** and **COMPANY.Sales**. The asserted relationship between **COMPETITOR** and **COMPANY** directs the reconciliation process to examine the relationships between the attributes of those objects. The process of conflict detection at the attribute level follows a pattern similar to that we have outlined at the object level: exploration, term expansion, conflict assertion and corroboration. Similarly, the corroboration process gathers knowledge through the invocation of reconciliation techniques, downward propagation, and the exploitation of any knowledge or metadata that has previously been acquired.

We present here the more interesting cases of the attribute comparisons likely to be investigated:

- **COMPETITOR.Name and COMPANY.Co-Name.**
  The attribute **COMPANY.Co-Name** does not match against any term in the query. *Term Expansion* through naming relationships, in this case the technique of Lexical Analysis for instance, would be applied to the term "Co-Name". Let us assume that the term expansion rendered the term "Name" as a synonym for "Co-Name". A correspondence could then be asserted between **COMPETITOR.Name** and **COMPANY.Co-Name**:

  Assert[COMPETITOR.Name,COMPANY.Co-Name]
  (synonym,class,(attribute,attribute))

  The fact that a correspondence was asserted between attributes of corresponding objects (**COMPETITOR** and **COMPANY**) provides additional corroboration to the assertion of a correspondence at the object level. Corroboration of these attributes will trigger more knowledge acquisition and further *downward propagation*, to the instance level.

- **COMPETITOR.Location** and **COMPANY.Location**
  COMPANY.Location can be asserted to be a synonym or a homonym of **COMPETITOR.Location**. Initially, both assertions are made, although the assertion of synonymy is a stronger assertion, since there already exists an assertion relating the objects **COMPETITOR** and **COMPANY**. That is, asserting a correspondence between **COMPETITOR.Location** and **COMPANY.Location** is similar to the process used to correspond the objects **COMPETITOR** and **COMPANY**; in the case of the attributes, however, that process is constrained by the prior knowledge that there is a relationship between **COMPETITOR** and **COMPANY**.

  We know that the two **Location** attributes are, in fact, homonyms, which contradicts the assertion of synonymy. *Contradiction* of the assertion of synonymy is achieved in the same way that *corroboration* is achieved, that is, knowledge is acquired, and the assertion is reviewed in context of that knowledge. For example, if descriptive metadata of the attributes are available, it will clarify that the attributes refer to different semantic concepts: **COMPETITOR.Location** refers to the names of headquarter cities while **COMPANY.Location** refers to the names of countries. If this does not resolve the conflict, reconciliation techniques trigger downward propagation. Through downward propagation, the instances of **COMPANY.Location** are collected, and compared to the instances of **COMPETITOR.Location**. This results in corroboration for the assertion that these attributes are homonyms (unrelated semantically), or perhaps an assertion that the attributes are related by aggregation. The query-directed approach of our methodology recognizes the aggregation relationship between the query term "Europe" and the term "country", and between the terms "country" and "city". Since "Europe" is a value of COMPETITOR.Mkt-Sgmt and not of COM PETITOR.Location, this triggers the comparison of COMPETITOR.Mkt-Sgmt and COMPANY.Location, which is discussed below.

- **COMPETITOR.Mkt-Sgmt and COMPANY.Location**
  The problem remains to establish a correspondence between **COMPETITOR.Mkt-Sgmt** and **COMPANY.Location**. As described above, the process of constraining the exploration by the query, called *Query-Driven Exploration*, results in the comparison of **COMPANY.Location** against **COMPETITOR.Mkt-Sgmt**. A correspondence can be asserted between **COMPETITOR.Mkt-Sgmt** and **COMPANY.Location** only if the term Mkt-Sgmt is recognized as a region or a location. Let us assume that metadata of the local schema for **Mkt-Sgmt** describes it as a region or a location. (Alternatively, lexical analysis, as well as knowledge of abbreviations is useful in parsing the term and analyzing its components.)

  Once a correspondence between **COMPETITOR.Mkt-Sgmt** and **COMPANY.Location** is asserted, the instances of **COMPANY.Location** are compared against the relevant instances of **COMPETITOR.Mkt-Sgmt** ("Europe"). These instances had been previously collected previously in an attempt to verify the correspondence between the two "Location" attributes. Recognition of the aggregation relationship results in the assertion

  Assert[COMPETITOR.Mkt-Sgmt,COMPANY.Location]
  (synonym,aggregation,(attribute,attribute))

  As the specific instances are discovered, an assertion is made for each instance of a country that is part-of Europe.

- **COMPETITOR.Qtly-Income and COMPANY.Sales**
  There is also no match in the target database for the attribute **Qtly-Income**. However, the attributes of **COMPANY**, previously collected, include an attribute **Sales**. Domain-specific knowledge bases exist (Sciore, et al., 1993) to supply translation or conversion functions between attributes. Such a knowledge tool, or, similarly, an expert system, is useful to recognize the correlation between **COMPANY.Sales** and

COMPETITOR.Qtly- Income, thereby expanding the **Sales** attribute through abstraction. (Recall that we include computed or derived functions as part of abstraction relationships.)

3.5.2. Mapping to Target Database #2

The problem of very different schemas. In examining Target database #2, we do not focus on the process of exploration through term expansion, which has been demonstrated in the discussion on Target database #1. Exploration through term expansion is required even for fairly similar schemas with slight naming and abstraction discrepancies in the representation of similar objects. This was the case of Target database #1. Here, we focus on the types of knowledge that are needed to detect and solve those conflicts that are due to very different schemas. As illustrated in the case of Target database #2, schemas can differ both in the conceptual model and in the data model. The exploitation of knowledge that exists in the metadata levels of heterogeneity plays a crucial role in detecting and resolving the semantic conflicts that are attributed to conceptual model and data model differences. Specifically, we discuss the database-level metadata that can assist in the detection of conflicts due to conceptual model differences. We also demonstrate that the knowledge of data model semantics can allow intelligent navigation during the reconciliation process, and can be useful in resolving conflicts due to data model differences.

Conceptual model heterogeneity. Target database #2 has a very different semantic emphasis than the local database: it is not organized with respect to companies; rather, Target database #2 maintains information about products. This emphasis is reflected in its schema, which is organized to store and access different products. The differences in the nature of the information that is stored in the databases may preclude reconciliation. For instance, if Target Database #2 did not have the object **PRODUCER-MNFCTR** as a property of the object **PRODUCT**, then it would be impossible to map **COMPETITOR** to any class of elements in Target Database #2 (assuming that there are no additional relations in the local database storing information about products). In our example, **PRODUCER-MNFCTR** is represented in the target schema, but does not include the attributes of interest to the local database. This conflict is a natural consequence of the different semantic emphases of the two databases, which are reflected in their conceptual models. Conceptual model differences between these two databases can impede reconciliation, even if Target database #2 was organized according to the relational data model.

Data model heterogeneity as a complicating factor. The difficulties in detecting conflicts which are due to differences in the nature of the information being stored is, in this case, exacerbated by the organization of the data according to different data models. For example, let us assume that the initial contact with Target database #2 identifies this system as an object-oriented database that stores information about products in the European Community. Local database metadata, in combination with term expansion techniques, as described above, supplies the needed information that the local database is owned by a producer of cold cereal "products". Let us assume that the *anchor* that is eventually found in the target database is the object **COLD-CEREAL**. At this point, reconciliation techniques begin requesting additional attributes of that object, as we have previously shown in the case of Target database #1. Many of these attributes are defined in the more general classes of which *COLD-CEREAL* is a specialization, that is, they are inherited by the object *COLD-CEREAL*. Thus, we require knowledge about the object- oriented data model in order to recognize the hierarchy of Product$\Longrightarrow$Food-Product $\Longrightarrow$Cereals$\Longrightarrow$Cold-Cereal. This understanding is needed to build a view of the object- oriented schema, and to ask appropriate questions of Target database #2 about the attributes of **COLD-CEREAL**. In other words, if a reconciliation technique triggers *downward propagation* by requesting additional attributes, knowledge of the data model must map that request into the appropriate traversal of the object-oriented schema. The reconciliation process can expect to encounter generalization hierarchies and must be equipped to ask Target DB #2 questions such as *"Are there any specializations of* **COLD-CEREAL**?" and *"What are the other subclasses of* **CEREAL**?" as well as *"What properties are inherited from the more general classes CEREAL, FOOD-PRODUCT,* etc. ?". Similarly, the corresponding object to **COMPETITOR** is the object **PRODUCER-MNFCTR**, which is also a property of **PRODUCT**. When the schematic relationship between **PRODUCT** and **PRODUCER-MNFCTR** is recognized, data model metadata will help clarify the semantic relationship between **PRODUCER-MNFCTR** and **COLD-CEREAL**. Ultimately, a search of **PRODUCT**, constrained by a set of **PRODUCER-MNFCTR** instances, will define the corresponding object of **COMPETITOR**.

Clearly, *database-level metadata* is required to expand query terms through our environmental knowledge. For instance, the knowledge that the local database is a cold cereal manufacturer is not explicitly represented in the schema, and cannot be acquired if upward propagation is strictly limited to examination of the schematic elements. *Data model metadata* is similarly clearly required in order to navigate and to understand the semantic relationships among the classes and subclasses of the object-oriented database. The interplay between database metadata and data model metadata can be seen in the different representations of **COMPETITOR** and **PRODUCER-MNFCTR**. Partially because the structure of the object- oriented schema, many of the attributes of **COMPETITOR** are not properties of the object **PRODUCER- MNFCTR**, which is itself, more properly, a property of **PRODUCT**. This can be understood within the context of the differences between the databases' contents, as well as their different data models. As we have shown, metadata at both the database and data models levels is needed to provide the additional knowledge that cannot be gained through simple schematic navigation or query term expansion.

3.5.3. Observations from the Mapping Process

We summarize the important points to be learned about the knowledge acquisition process from the mappings discussed above.

1. The type of knowledge that was needed, in all cases, was knowledge of naming and abstraction relationships, as well as schematic knowledge in order to traverse the levels of heterogeneity.

2. The extent of the automation that is possible is limited by the techniques available. For example, understanding that competitor must be mapped by the attribute COMPANY.Industry may require human intervention, unless sophisticated Knowledge Sources are available.

3. The possibility of successful reconciliation is limited by the knowledge available, regardless of whether the source of that knowledge is human or automatic. For example, the ability to estimate quarterly income for a particular industry in a specific region may require expert knowledge. The sources of knowledge can include schematic knowledge native to the reconciler, various reconciliation and exploration techniques (such as object identification techniques), as well as metadata, domain-specific knowledge bases, concept hierarchies, lexical analyzers, thesauruses or expert knowledge.

4. Ordering is inherent in mapping the elements between the local and target databases. For example, until **COMPETITOR** and **COMPANY** are asserted to be corresponding objects, the attributes of **COMPANY** will not be examined, and subsequent correspondences cannot be asserted. This ordering is achieved through the upward and downward propagation through levels of heterogeneity.

5. Similarly, there is a recurrent pattern in the context building process, across levels of heterogeneity. If exploration of the target database is unsuccessful, context is expanded through query term expansion. These expanded terms can then be compared against the knowledge that has been acquired thus far, and can trigger further exploration. If exploration is successful in finding an anchor in the target database, that anchor is compared, and, if necessary, expanded through term expansion. Schematic knowledge is exploited in navigating up and down the levels of heterogeneity, as exploration and reconciliation techniques seek more knowledge. This knowledge may be in the form of more anchors, in the case of exploration, or in corroborating evidence, in the case of reconciliation techniques. The interplay of exploration, context expansion or constraining, assertion of inter-schema correspondences, invocation of reconciliation techniques, gathering of evidence to corroborate and contradict assertions, and navigation through the levels of heterogeneity is evident throughout the conflicts that have been illustrated in this example.

6. The reconciliation process requires both *positive evidence* in order to corroborate assertions and *negative evidence* in order to contradict assertions. For example, the correspondence between **COMPETITOR.Name** and **COMPANY.Co-Name** is corroborated through the positive evidence of the similarities at the instance level. On the other hand, the assertion of a synonymous relationship between **COMPETITOR.Location** and **COMPANY.Location** is contradicted through the negative evidence of the metadata describing those attributes.

7. Target database #1 involves an implicitly simple case in which the relevant elements are located in one relation in each database. Nevertheless, even in this relatively simple case, explicit knowledge is required of synonyms, homonyms, generalization, functional relationships, schematic metadata, schematic structure, expert knowledge and standard information retrieval techniques.

8. Exploration and knowledge acquisition are query directed, that is, the query constrains the next step of the reconciliation process. For example, the aggregation relationship between "Europe" and "country" triggers a comparison between **COMPETITOR. Mkt-Sgmt** and **COMPANY.Location**.

9. Context building accumulates knowledge and exploits previously acquired knowledge as new conflicts are encountered. For example, the instances of **COMPANY.Location** were collected in order to compare that attribute to **COMPETITOR.Location**. These instances were later used in order to compare **COMPETI TOR. Mkt-Sgmt** to **COMPANY.Location**.

10. The number of cases examined in this example is less than the twelve possibilities outlined by the classification. For example, in the case of **COMPETITOR** and **COMPANY**, there are only five possible assertions, of which only the three most likely are actively pursued. Discrimination among the possible assertions, and determination of the most likely assertions are discussed further in the next subsection.

4. Context Management using a Truth Maintenance System

Plausible ISCAs: Incompleteness and uncertainty. In an incremental system, assertions are made under conditions of partial knowledge. Referring to our example of **COMPETITOR.Location** and **COMPANY.Location**, in the absence of contextual knowledge, we could reasonably propose that the attributes are synonyms or that they are homonyms. These two assertions are contradictory to each other, yet they are temporarily both compatible with the evidence (or lack of it), until further evidence (knowledge) is gathered. As we gradually acquire new knowledge, it may serve to *corroborate* or *contradict* one or the other of those assertions. As the evidence mounts, our belief in favor of some assertions grows stronger. Thus, the assurance that there exists support for a given assertion or a set of assertions take various strengths, depending on the strength of evidence available. This degree of assurance is referred to here as the *degree of belief*. In our example of mapping to target database #1, the examination of the set of instances **COMPANY.Location** provides knowledge, which decreases our degree of belief in the assertion that the attributes are synonyms, and also increases our degree of belief in the assertion that the attributes are homonyms. The function of the knowledge acquisition process is to elicit evidence with which to resolve conflicts with incremental certainty.

Clearly, the nature of semantic reconciliation imposes on us to seek techniques that can be used to reason effectively when complete, consistent and constant model of the world is not available. Most approaches in heterogeneous databases circumvent the complexity of the problem either by imposing a global schema, where in effect all the conflicts are resolved, or by designing techniques which only work in a very specific environment. The implication is that the model of reconciliation is completely specified.

We use the Dempster-Shafer (D-S) theory (Dempster, 1968; Shafer, 1976) to compute the degree of belief that is warranted given the evidence. Unlike pure Bayesian theory, this approach does not require specification of a complete probabilistic model before reasoning can commence. Further, it is compatible with the classical, proof-based style of logical inference, sharing the syntax of deductive databases and logic programming. In other words, it can be used in conjunction with rule-based systems. An interesting aspect of the D-S theory is that it is domain-independent. It provides a mechanism to infer degrees of belief without regard to the nature of the evidence. Below, we discuss briefly how this theory is used in our application and we define along the way the useful concepts.

Note that the bayesian approach may be appropriate in some cases. For instance, Chatterjee and Segev (Chatterjee and Segev, 1993) use it successfully in designing a framework where rule based probabilistic join operators can be handled. An important aspect of this technique is that equivalence of tuples in the two relations considered may be determined semantically. However, the framework is developed under a number of assumptions which restrict its applicability. It assumes that the two relations have already been determined to be equivalent on the basis of that the attributes are equivalent. As shown in (Lim, et al., 1993), this cannot be guaranteed in general. Further, the method does not show how semantic equivalences between tuples can be discovered, although it suggests a way of representing semantics using the idea of semantic domain vector. The method attempts to determine the best match between the tuples, assuming the equivalence of tuples. Obviously, this paradigm is not appropriate to the general knowledge acquisition problem addressed here.

Assigning degrees of beliefs to assertion: The D-S approach considers a set of assertions and assigns to each one of them, say assertion A, an interval $[B, Pl]$ in which the degree of belief must lie. B measures the strength of evidence in favor of a set of assertions. It ranges from 0 (no evidence) to 1 (certainty). Plausibility is defined to be: $Pl = 1 - \bar{B}$ also ranges from 0 to 1 and measures the probability that an assertion A is compatible with the evidence, i.e., the probability that it cannot be disproved and is therefore possible. $\bar{B}$ measures the strength of evidence in favor of (not A).

Consider again the relationship between COMPETITOR and COMPANY of our previous example. Per our classification, there is a set of twelve possible assertions that can be made. Let this set be Ω. In the absence of any information, the true likelihood of each assertion in Ω lies in the range $[0, 1]$ according to the D-S theory. As evidence is accumulated, this interval is expected to shrink representing increasing confidence that the assertion can be supported. So far, we have talked intuitively about the degree of our belief in some assertion given some evidence. Let us now define it more precisely. We start with the exhaustive universe of initially exclusive assertions Ω. This is referred to as the *frame of discernment* in the D-S theory. Our goal is to attach some degree of belief of Ω. However, not all evidence is directly supportive of individual elements. Generally it supports subsets of the frame of discernment. For example, the fact that the following evidence was uncovered:

E1: There is a terminological generalization relationship between "COMPANY" and "COMPETITOR".

supports a subset of Ω. In addition, since the elements of Ω are mutually exclusive, evidence in favor of some may have an effect on our belief in the others.

The DS theory defines a probability density function, which we denote d. The function d is defined not just for individual assertions in Ω but for all subsets. The quantity d(q) measures the amount of belief that is currently assigned to exactly the set q of assertions. Since Ω contains in our case 12 elements, then there are 2^{12} subsets of Ω. We must assign d so that the sum of all degrees of belief of all subsets of Ω is 1. Although dealing with 2^{12} values for each pair of concepts X and Y may appear intractable, it usually turns out that many of the subsets will never need to be considered because they have no significance in the problem domain and so their associated degree of belief d will be 0.

Let us now see how degree of belief d works in our example. Assume that we have no information which will permit us to select among the 12 assertions when we start our reconciliation process. Then we define d as:

$$\{\Omega\} \quad (1.0) \qquad (1)$$

That means that the degree of belief d in any of the subsets of Ω is 0. Although the actual value must be one of the individual assertions in Ω, we do not have any information that allows us to discriminate among the assertions. This disambiguation can be achieved by passing evidence E1 to reconciliation techniques, such as those in (Li and Clifton, 1993), and/or to our own local knowledge sources, to possibly derive relationships between objects "COMPETITOR" and "COMPANY". Before we proceed with our example, however, we briefly digress to describe how our system interfaces with these techniques and knowledge sources. Basically, the interface consists of the following template:

r: IF C(*m*) THEN consequent. [*m.m*']

where the antecedent of the rule C is defined recursively in BNF as follows:

C ::= E | Assertion | Assumption | E and C | Assertion and C| Assumption and C.

where "E" denotes a given piece of evidence such as E1, "Assertion" indicates an assertion previously derived, "Assumption" represents a piece of knowledge at the same level of granularity as E which needs to be corroborated, "consequent" is a disjunction of assertions about two objects O1 and O2, *m* represents the degree of belief in all the assertions in C, and finally *m'* is the degree of belief in rule r if m=1. Using the *series reduction* operation in rule networks, *m* is the product of all the degrees of belief in C. In the event, there are several rules to derive the consequent of rule r, the D-S theory provides a feature called *parallel reduction* to derive the degree of belief in the consequent. However, all the matching rules need not be fired at the same time. Strategies need to be designed to select among all the relevant rules.

This type of rules is referred to as **premises** in the D-S theory. The degree of belief could have been assigned to these premises either by reconciliation techniques such as Lim et al. similarity measure (Lim, et al., 1993), or elicited from experts in integration or synthesized from our own experience using our system. It is, however, beyond the scope of this paper to discuss this aspect of our research. As mentioned previously, assumptions in the antecedent of the rule represent abstraction or naming relationships whose validity can only be confirmed by seeking evidence in their favor. In the meantime, these assumptions could be made and then used to drive the truth maintenance system discussed later in this section. We now resume our example. Let us assume that evidence E1 matches the following rule:

r1: IF gen(T1,T2) THEN gen(O1,O2) or class(O1,O2). [0.6]

where gen(T1,T2) denotes a generalization between term T1 and T2, while gen(O1,O2) and class(O1,O2) denote a generalization and a class relationship, respectively, between the objects corresponding to terms T1 and T2. Obviously, rule r1 fits the template described above. Rule r1 permits us to conclude that, with a confidence level of d_1=0.6, the set Ω_1 of plausible assertions is:

A1: **Assert**[COMPETITOR,COMPANY](synonym,class,schema)
A2: **Assert**[COMPETITOR,COMPANY](synonym,specialization,schema)
A3: **Assert**[COMPETITOR,COMPANY](homonym,class,schema)

We update d as follows:

$$\begin{array}{rl} \{\Omega_1\} & (0.6) \\ \{\Omega\} & (0.4) \end{array} \tag{2}$$

At this point we have assigned to the set Ω_1 the appropriate degree of belief. The remainder of our belief still resides in the larger set Ω. Obviously, the set of plausible assertions was determined by assuming that the additional conditions C beyond evidence E1 in the antecedent of rule r1 are true with some degree of belief. This assumption allows us to resume reconciliation, while prompting our knowledge acquisition process to gather evidence to corroborate or contradict the validity of these conditions.

The acquisition of even a single piece of evidence has the potential of reducing significantly the number of possible assertions that can be made. For instance, evidence E1 guarantees that the only other possible assertions, besides those listed above, are:

A4: **Assert**[COMPETITOR,COMPANY](synonym,part-of, (object,object))
A5: **Assert**[COMPETITOR,COMPANY](synonym,function, (object,object))

In this case, the number of possible assertions is reduced from 12 to 5. The number of subsets is reduced from 2^{12} to 2^5. Clearly, this is a significant reduction.

As illustrated in our mapping to target database #1, the knowledge acquisition process discovered, through downward propagation, the following evidence:

E2: Terms "Name" and "Co-Name" are synonyms.

Our objective is to illustrate how the combination of evidence E1 and E2 constrains the set of likely assertions. Let us assume that passing evidence E2 to reconciliation techniques and other knowledge sources resulted in the derivation of the following assertion between attributes ' 'Name" and "Co- Name" of COMPETITOR and COMPANY:

Assert[COMPETITOR.Co-Name,COMPANY.Name]
(synonym,class,(attribute,attribute) [0.9]

Let us assume that this assertion, when passed to reconciliation techniques and/or our local knowledge sources, triggers the following rule:

```
r2:  IF Assert[att1,att2](synonym,class(attribute,attribute))(0.9)
        AND Key(att1) AND Key(att2)
     THEN Assert[O1,O2] (synonym,generalization,(object,object))
        OR Assert[O1,O2](synonym,class, (object,object)) [0.8=0.88*0.9]
```

where att1 and att2 denote attributes of object O1 and O2, and Key(att1) and Key(att2) denote that att1 and att2 are candidate keys of O1 and O2. We assume that the degree of belief in rule r2 was deduced from the degree of beliefs in the antecedent. In this case, we assume that the degrees of belief in Key(att1) and Key(att2) are close to 1. Obviously, rule r2 matches the template given in rule r. Key(att1) and Key(Att2) represent those assumptions which will be used to drive the truth maintenance system discussed later.

¿From rule r2, we can deduce, with a level of confidence d_2=0.8, that the set of plausible assertions Ω_2 for the relationship between COMPETITOR and COMPANY consists of A1 and A2. In other words, we have:

$$\begin{array}{ll} \{\Omega_2\} & (0.8) \\ \{\Omega\} & (0.2) \end{array} \qquad (3)$$

We have now set up a situation which we can use to show how two pieces of evidence are used in the D-S theory to constrain the set of plausible assertions. Observe that

Table 1. New frames of discernments.

		Ω_2	(0.8)	Ω	(0.2)
Ω_1	(0.6)	{A1,A2}	(0.48)	Ω_1	(0.12)
Ω	(0.4)	Ω_2	(0.32)	Ω	(0.08)

assertions A1 and A2 appear in both Ω_1 and Ω_2. Therefore, our degree of belief in these two assertions should be altered by combining the two pieces of evidence. The D-S theory provides a simple mechanism for computing the degrees of belief of all intersections of current frames of discernment. It defines the degree of belief d_3 as follows:

$$d_3(S) = \frac{\sum_{S1 \cap S2 = S} d_1(S1).d_2(S2)}{1 - \sum_{S1 \cap S2 = \emptyset} d_1(S1).d_2(S2)} \tag{4}$$

With this formula, we can derive new frames of discernment with their degrees of belief as shown in table 1.

The above example provides a brief presentation on how the Demspter-Shafer theory is used to assign degrees of belief to derived frames of discernments. There are a number of issues, concerning the D-S theory, which are beyond the scope of this paper. For example, a careless use of this theory may cause reasoning to inherit many problems associated with monotonic logic.

Assigning logical values. The use of the D-S technique requires an inference engine to deduce belief functions. We use a truth maintenance system to provide a symbolic mechanism for identifying the set of assumptions needed to assemble the desired proofs, so when we assign probabilities to these assumptions, the system can be used as symbolic engines for computing the degrees of belief sought by the D-S theory. The second important use of the truth maintenance system is to handle the effect of retracting assumptions when they are invalidated by the evidence and to keep track of the multiple plausible sets of assertion which can coexist in the absence of complete knowledge. Truth maintenance systems arose as a way of providing the ability to do dependency-backtracking when assumptions or assertions, such as the ones discussed above, are retracted because they are contradicted by the current knowledge, and so to support nonmonotonic reasoning. Nonmonotonic reasoning is an approach in which axioms and/or rules of inference are extended to make it possible to reason with incomplete information. At any given moment, in systems based on this approach, an assertion is either believed to be true, in our terminology *Confirmed*, believed to be false, in our case *Retracted*, or not believed to be either, in our case *Undetermined*. How then are these values assigned in our system? A value *Undetermined* is assigned when, based on the current evidence, A and not A coexist in the considered set of assertions. A value *Retracted* is assigned when an assertion is not under consideration either temporarily if its degree of belief is greater than 0 or permanently if its degree of belief is 0. Finally, *Confirmed* is assigned if its degree of belief is greater than 0, and its contradiction is not under consideration.

Consider again our example Ω which consists of assertions A1, A2,A3,A4 and A5. Initially, A4 and A5 were at least temporarily eliminated from consideration; therefore their logical value is *Retracted.* The remainder, Ω_1, which consists of A1, A2 and A3, is assigned values *Undetermined,* since in this case A1 and its contradiction coexists.

After progressing in the process of reconciliation, we get Ω_2 which consists of the two non-contradicting assertions A1 and A2. Therefore, these assertions are assigned values *Confirmed.* The propagation of this value is performed, in our application, by an Assumption-Based Truth Maintenance System (ATMS) (DeKleer, 1986).

Multiple Context Management: A **context** is a set of consistent assertions. Therefore, these assertions must be all *Confirmed.* An *Undetermined* may be part of a context so long as its contradiction is not. In our example, conflicting assertions will be generated describing the correspondence of the **Location** attributes. The assertion of synonymy and the assertion of homonymy are inconsistent, and cannot simultaneously occur within a single context. Therefore, several contexts may co-exist at any given point in time, until this is disambiguated by further information. The sets of assertions in each context may be disjoint or intersecting. Each set of assertions is managed by an ATMS which provides a mechanism for keeping track of a context of noncontradictory plausible assertions, and the evidence developed during an inference session. Plausible assertions are built largely on default assumptions and educated guesses, and an ATMS is able to retract some of these assumptions in light of new information. All conclusions that were derived from these assumptions have to be retracted as well. The task of reconciliation is to validate or invalidate contexts as more knowledge is acquired. Since there may be more than one context maintained at any one time, we select the most likely context for further refinement based on the degrees of belief of the assertions in that context.

Soundness and Completeness. We believe that the incremental building of context does not support the concepts of *soundness* or *completeness* of context. By soundness, we refer to the capability to infer correctly and with certainty the ISCAs between concepts, that is, a retracted or a confirmed assertion with degree of belief 0 or 1, respectively. By completeness, we refer to the capability to infer every sound ISCA, that is, if the ATMS returns a confirmed value with degree of belief 1 or a retracted value with degree of belief 0 for every plausible assertion. Even the "confirmed" ISCAs cannot be proven to be sound. Rather, all ISCAs are merely plausible, with different levels of evidence to support them. Thus, we emphasize a context that is *satisficing* with respect to the reconciliation task at hand. Satisficing is measured by how close we are to finding a context in which the query can be answered. In other words, how close we are to finding a complete proof for our context. SCOPES depends heavily on knowledge bases in order to heuristically determine a "good enough" level of confidence to provide a "good enough" response to a query. Unlike monotonic systems, ATMS does not dispose of the inferences, once the conclusion was established. On the contrary, by recording the history of inferences, this system can retrace the source of beliefs, a feature necessary for generating explanations and for resolving contradictions.

5. SCOPES

5.1. Objectives of SCOPES

The primary objective of SCOPES is to coordinate the reconciliation process by building a context within which meaningful communication with other systems can take place. SCOPES builds context by acquiring semantic knowledge. Below, we briefly introduce the SCOPES architecture, listing the components that are required for interoperability and context building. We then discuss the design properties of SCOPES that support dynamic integration. We propose a coordination algorithm that describes the coordination necessary among the components of SCOPES. Finally, we discuss the complexity issues of the coordination algorithm.

5.2. The SCOPES Architecture

The architecture of SCOPES is illustrated in Figure 2. SCOPES consists of one or more Coordinators, working in parallel under the supervision of a SuperCoordinator. The objective of each Coordinator is to build a context with one other system (a *binary context*), which is then forwarded to the SuperCoordinator.

Each coordinator has the following components:

1. **Dialoguer.** The Dialoguer consists of two sub- components: the Semantic Communication Protocol (SCP) and the Request Translator (RT). The SCP is a protocol to handle communication with another system. It is modelled after the Conversation Architecture proposed by Winograd in (Winograd and Flores, 1986). The Dialoguer is responsible for establishing contact, and initializing the parameters of the dialogue. Arrow #1 represents the communication between the Coordinator and the Dialoguer. The information that is requested is translated by the Request Translator into requests that can be understood by the external system.

2. **Context Builder.** The function of the Context Builder is to build context assertions about another system. The Context Builder has two components: the Conflict Detector, which detects and classifies semantic conflicts; and the Integrator, which uses integration rules to integrate inter-schema assertions into existing context assertions, and which organizes context assertions by level of heterogeneity. Arrow #2 represents the communication between the Coordinator and the Context Builder. As conflicts are identified, the Context Builder sends them to the Coordinator, which triggers the appropriate Reconciliation Techniques. We have described the maintenance of multiple contexts in the previous section.

3. **Reconciliation/Exploration Techniques.** The Reconciliation Techniques return to the Coordinator either one or more proposed conflict resolutions, in the form of inter-schema correspondence assertions, or requests for more information. The communication between the Coordinator and the Reconciliation Techniques is represented by multiple arrows #3. The Reconciliation Techniques also access Knowledge Sources,

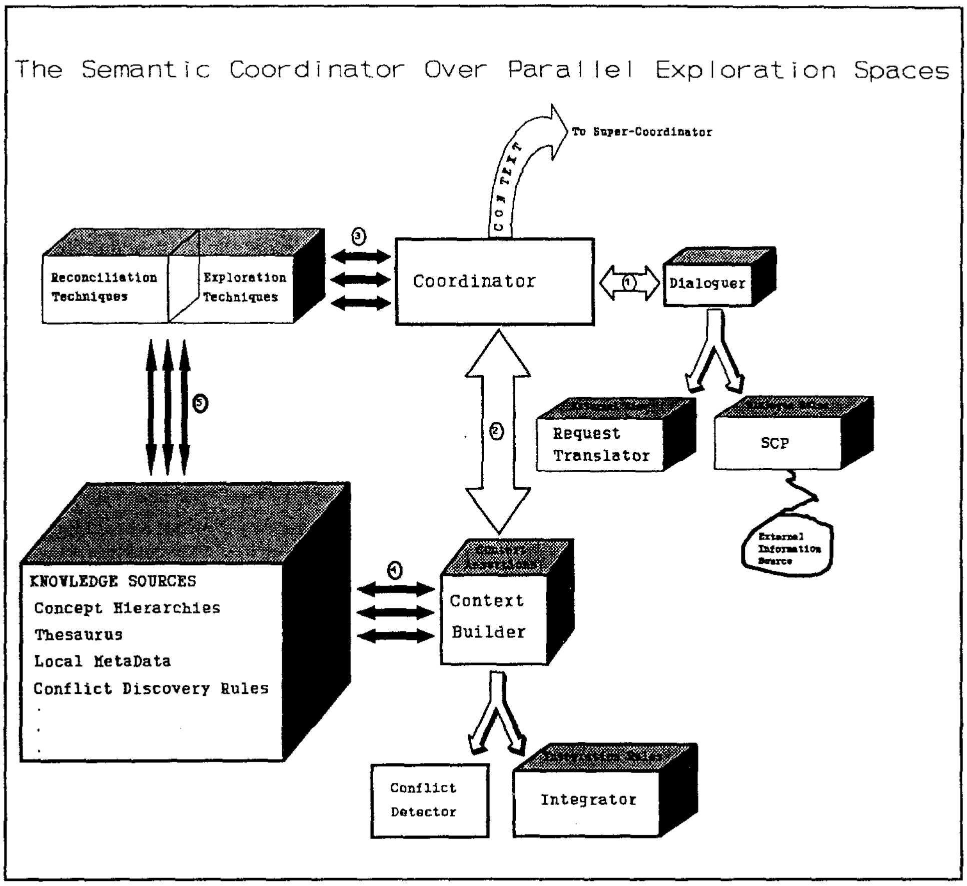

Figure 2. The SCOPES Coordinator

such as a thesaurus, etc. (multiple arrows #5) or previous knowledge. Exploration Techniques are similar in function to Reconciliation Techniques. However, they are initiated by the Coordinator only when other attempts at context building have been exhausted, and yet the context remains insufficient to answer the query.

4. **Knowledge Sources.** Reconciliation is a knowledge- intensive process. Knowledge of the required naming, abstraction and schematic relationships is contained in schematic information as well as in descriptive environmental metadata.

5.3. Dynamic Reconciliation Properties of SCOPES

- *Computer-Aided Reconciliation.* The incremental discovery approach used by SCOPES results in a more flexible system, as the system can dynamically determine the next step, depending upon the knowledge that is available. The coordination of different reconciliation techniques supports semantic communication independence, and also increases automation, thereby supporting the goal of transparency.

- *Binary Context Building.* SCOPES builds one context for each system with which it communicates; binary context building reduces the problem of scalability, and supports extensibility.

  Learning. A structure of past interactions provides some stability for later exchanges by eliminating the necessity for the databases to continuously and spontaneously reinvent their reconciliation when in the presence of new queries. Memories stored in a given context can provide the basis of future interaction, if the knowledge acquired in past interactions is be validated again, in case updates occurred in the meantime.

  Parallelism. The multiple arrows (3, 4 and 5) of Figure 2 illustrate the parallelism supported by the SCOPES architecture. Several Reconciliation and Exploration Techniques may be activated concurrently. The techniques are also decomposed into their component subtasks, which can operate in parallel, each producing inter-schema correspondence assertions, which may be handled concurrently by the Context Builder.

  Exploration. Hewitt (Hewitt, 1985) describes the process of exploration, emphasizing the open-ended nature of a non- goal-directed search. SCOPES differs from this approach in that the query imposes a scope on the search, by restricting the exploration to those terms that are present in the query, or by triggering the expansion of those terms through naming, abstraction or schematic relationships.

6. The Coordination Algorithm

In (Ouksel and Naiman, 1992), we presented those aspects of the coordination algorithm that illustrated how precedence relationships are dynamically determined in this approach. Here, we present the complete coordination algorithm to show the coordination of the

```
Initialization--handled by Supercoordinator
    A. Analyze query to compile a list of query terms
    B. Initialize Dialogue with Remote Database
        1. Policy Handshake
            a. Identification
            b. Authorization
            c. Security
            d. Availability of Export Schema
            e. Availability of Semantic Information
        2. Semantic Handshake--Database Relevancy
            a. Criterion: identification
            b. Criterion: other DBs available
            c. Criterion: examination of export schema (if any)
            d. Criterion: performance
        3. Syntactic Handshake
            a. Data Model
            b. Build view of export schema (if any)
```

Figure 3a. Initialization

different components of the SCOPES architecture. We briefly describe the main steps of the algorithm.

6.1. Initialization

The initialization of the coordination algorithm is shown is Figure 3a. The Dialoguer is responsible for establishing contact, and initializing the parameters of the dialogue. Initialization includes policy, semantic and syntactic "handshakes" with the remote database.

The *policy handshake* is the exchange of identification descriptors, as well as the authorization and security constraints on the exchange of information. If an export schema is available, it is solicited. In addition, the level of semantic information to be exchanged must be established. For example, if a term is present in the remote database as an attribute, must the local database request the associated object, or will this information be automatically supplied? Will the related semantic metadata be automatically supplied, or will it be available on request? The purpose of the *semantic handshake* is to determine the relevance of the remote database to the query. In our example, Target Database #1, which maintains information about companies, is more relevant to the query than Target Database #2, which maintains information about products. Both of the target databases are more relevant than, for instance, a database maintaining medical records.

The *semantic handshake* requires identification of the re mote database, which is forwarded to the Supercoordinator. The Supercoordinator can rank the relevancy of all of the available remote databases, similarly to the way that documents are ranked for relevancy by Information Retrieval techniques (Salton, 1989). Determining relevancy is made easier if an export schema is available. The decision to continue a dialogue with a specific database is based on the relevancy ranking as well as performance considerations. The Request Translator requires knowledge of the remote database's data model in order to properly formulate questions to the remote database, as well as to build a view of its schema. Semantic enrichment of less expressive data models, as proposed in (Castellanos, et al., 1992) for example, can be useful in this translation. The exchange of data model knowledge is the *syntactic handshake*.

6.2. The Body of the Coordination Algorithm

The Coordination Algorithm is shown in Figure 3b. In discussing the main steps of the algorithm, we shall refer to our example.

I. Initiate Dialogue. This step describes the exploration of the external system. In our example, the dialogue included *term exploration*, for instance, the attempt to match the query term **COMPETITOR**; *schematic queries*, for example, all instances of **COMPANY.Location**; and *metadata requests*, for example, the metadata describing the attribute **COMPANY.Location**.

II. Build context as a result of external exploration. In our example, we infer plausible ISCAs of synonymy and homonymy between the two **Location** attributes, after exploration of Target Database #1 returns **COMPANY.Location** and the term matches against a query-related attribute. These assertions *trigger* internal and external exploration requests, in order to gather evidence for their corroboration or contradiction. The assertions also propagate through context, by providing corroborating evidence of the correspondence between **COMPETITOR** and **COMPANY**. (Propagation of assertions through levels of heterogeneity, and the management of multiple contexts through the use of the D-S techniques, has been discussed in the previous section.) As a result of the propagations and the context management, further internal and external exploration requests are triggered.

III. Compile internal triggers. In our example, term expansion of the query term **COMPETITOR** triggers the invocation of the internal naming and abstraction knowledge sources. As discussed earlier, object identification Reconciliation Techniques are also triggered, to determine, for example, what information might be needed to confirm a correspondence between **COMPETITOR** and **COMPANY**. All of the triggers of internal knowledge sources are compiled at this point. (Compilation and execution of internal triggers is done before further dialogue with the remote systems, in order to exploit the knowledge that has been acquired from previous iterations of dialogue.)

IV. Schedule and execute all internal triggers. Checking of synonyms, abstractions, lexical analysis, etc. should only be done once, and the results applied wherever needed. Decomposed Reconciliation Techniques may require sequential operation, or may be invoked in parallel, as discussed previously in the section on parallelism. Processing is scheduled for those reconciliation techniques that will further the most likely context.

V. Build context as a result of internal exploration. This step includes the assertion of ISCAs, and their propagation through context, just as in **Step II**. The only exception

```
Loop: While (query not yet answered and further dialogue is possible)

I. Initiate Dialogue.
   A. Send list of requests from remote DB to Request Translator for translation to
      remote DB's Data Model. The list of possible requests consists of:
      1. Term exploration
      2. Schematic queries
      3. Metadata queries
   B. Export schema, if available, is considered the first round of dialogue.
II. Context building as a result of external exploration.
   A. Infer plausible ISCAs from responses
      1. Generate plausible ISCAs from the discovered correspondences.
      2. Trigger internal term expansion for unmatched terms.
   B. Propagate the plausible ISCAs as assumptions in the ATMS.
      1. Propagate upwards/downwards through the levelof heterogeneity.
      2. Propagate through Context. Detect conflicts. Modify the degrees
         of belief based on this new corroborating or contradictory
         evidence. Generate or eliminate contexts.
III. Compile "internal" exploration (Reconciliation Techniques)
     triggers. Corroborate or contradict ISCAs.
IV.  Schedule and execute all "internal" triggers. Schedule parallel
     and sequential operations. Apply prioritization heuristics. Execute
     internal triggers.
V.   Context building as a result of internal exploration.Analogous to
     Step II above.
VI.  Compile "external" exploration (Dialogue) triggers. Trigger
     possible dialogue directly from ISCAs. Trigger possible dialogue
     with the expanded search terms (to find new anchors).
VII. Prioritize dialogue with remote database based on several criteria:
     preponderance of evidence, specificity, and other heuristics such
     as performance or effectiveness.

Endwhile.

Assuming no further term expansion of possible, then either the
reconciliation is sufficient to answer the query, or human intervention
is necessary to generate new plausible ISCAs or new terms
to continue the dialogue.
```

Figure 3b. The Coordination Algorithm

is that the source of the knowledge acquired was internal knowledge sources rather than exploration of the target database.

VI. Compile external exploration (Dialogue) triggers. In our example, some of the Reconciliation Techniques trigger, for instance, requests for the other attributes of the objects **COMPANY**. These requests are compiled, redundant requests are eliminated, etc., in preparation for the next round of dialogue.

VII. Prioritize dialogue with the external system. This is analogous to the scheduling of internal triggers (discussed in **Step IV**, above). In the case of communication with an external systems, consideration is given not only to issues of *effectiveness* (for instance, we pursue the most likely context), but also to issues of performance (for instance, by favoring a specific schematic query to a more general term exploration or a request for a complete set of instances).

The cycle of exploration and context building continues until reconciliation is achieved (i.e., all query terms have been mapped with confirmed assertions), or no further term expansion or possible dialogue is possible. In the first case, the reconciliation process can be successfully terminated; in the second case, the reconciliation process can be unsuccessfully terminated, or human intervention can be requested to provide new ISCAs.

7. Conclusion

We have presented a Semantic Coordinator Over Parallel Exploration Systems to coordinate the different processes needed to achieve interoperability. We have described the components of SCOPES, which exploit any currently available tools, to incrementally build the context for semantic reconciliation. flow. A number of advantages accrue from using SCOPES: (i) flexibility, (ii) scalability, (iii) semantic communication independence, and (iv) transparency as explained in (Ouksel and Naiman, 1992). The design of SCOPES raised a number of issues which require further elaboration. Below is a nonexhaustive list of issues we are currently investigating:

- The robustness of the classification needs further investigation to show whether all the various semantic conflicts discussed in the literature can be captured by a set of logically connected assertions.
- The assignment of a measure of belief to premises requires clarification. We are currently investigating how expert knowledge can be elicited or a reconciliation techniques are used to assign these initial values.
- We do not believe it is possible to design a deterministic mechanism for this task of comparing multiple contexts based on measures of belief. The only way, therefore, is to devise heuristics which will allow comparisons of these contexts. We are currently investigating such heuristics.
- We are currently investigating heuristics to constrain the growth of the search space by assigning logical values of *Confirmed* only when the degree of belief exceeds a threshold.

- The truth maintenance system provides a way of generating explanations on the steps that led to a specific reconciliation. This aspect requires however further exploration.
- Design of criteria on which to base termination of exploration.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments and Arnie Rosenthal for his insights on a preliminary version of this paper.

Notes

1. Supported by a grant from the Economic Development of Commission of the city of Chicago and a CBA 2000 award from the College of Business Administration at University of Illinois at Chicago.

References

Batini, C., Lenzerini, M. and Navathe, S. (1986). A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4), 1986.

Bertino, E., Negri, M., Pelagatti, G. and Sbattela, L. (1990). An object-oriented data model for distributed office applications. In *Proceedings of the Conference on Office Information Systems, MIT Cambridge, Mass.*, April.

Brodie, M. L. (1984). On the development of data models. In *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages edited by Michael L. Brodie, John Mylopoulos, Joachim Schmidt. New York: Springer- Verlag.*

Castellanos, M. (1993). Semantic enrichment of interoperable databases. In *RIDE-IMS '93 Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, April. Vienna, Austria.

Castellanos, M., Saltor, F. and Garcia-Solaco, M. (1992). A canonical model for the interoperability among object-oriented and relational databases.

Chatterjee, A. and Segev, A. (1993). Rule-based joins in heterogeneous databases. *Decision Support Systems.* to appear in Decision Support Systems.

Chomicki, J. and Litwin, W. (1992). Declarative definition of object-oriented multidatabase mappings. In *International Workshop on Distributed Object Management*, pages 307–325, August. Edmonton, Canada.

DeKleer, J. (1986). An assumption-based truth maintenance system. *Artificial Intelligence*, 28.

Dempster, A. P. (1968). A generalization of the bayesian inference. *Journal of the Royal Statistical Society, Series B*, 30:205–247.

Elmagarmid, A. and Pu, C. (1990). Introduction to the special issue on heterogeneous databases. *ACM Comput. Surv.*, 22.

Fang, D., Hammer, J. and McLeod, D. (1992). An approach to behavior sharing in federated database systems. In *Proc. of the International Workshop on Distributed Object Management* T. Ozsu, U. Systems and P. Valduriez editors Edmonton, Canada.

Garcia-Solaco, M., Castellanos, M. and Slator, F. (1993). Discovering interdatabase resemblance of classes for interoperable databases. In *RIDE-IMS '93 Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, April. Vienna, Austria.

Giladi, R. and Shoval, P. (1993). An intelligent system for routing user requests in a network of autonomus heterogeneous databases. In *Next Generation of Information Technology Systems*, June. Haifa, Israel.

Hewitt, C. (1985). The challenge of open systems. *BYTE*, pages 223–242, April.

Ioannidis, Y. E. and Livny, M. (1989). Data model mapper generators in observation dbmss. Dec. Chicago, Il.

Kent, W. (1979). The entity join. In *Proceedings of the Fifth International Conference on Very Large Databases*, pages 232–238, October.

Larson, J., Navathe, S. and Elmrasri, R. (1989). A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15:449–463.

Li, W. S. and Clifton, C. (1993). Using field specifications to determine attribute equivalence in heterogeneous databases. In *RIDE-IMS Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, April. Vienna Austria.

Lim, E. P., Srivastava, J., Prabhakar, S. and Richardson, J. (1993). Entity identification in database integration. In *Proceedings of the 8th International Conference on Data Engineering*, pages 294–301.

Litwin, W. (1989). A model for computer life. In *In the Workshop on Heterogeneous Database Systems* P. Scheuermann and C. Yu editors, Dec., Chicago, Il.

Litwin, W., Mark, I. and Roussoupoulos, N. (1990). Interoperability of multiple autonomous databases. *ACM Comput. Surv.*, 22(3), 1990.

Meng, W. (1992). A theory of translation from relational queries to hierarchical queries. *PhD Dissertation, University of Illinois at Chicago.*

Ouksel, A. and Naiman, C. (1994). A classification of semantic conflicts. sumitted to *Decision Support Systems.*

Ouksel, A. and Naiman, C. (1992). Semantic mechanisms for heterogeneous information systems. In *Workshop on Information Technology Systems*, Dec. Dallas, Texas.

Qian, X. (1993). Semantic interoperation via intelligent mediation. In *RIDE-IMS Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, April. Vienna Austria.

Ram, S. and Barkmeyer, E. (1991). A unifying semantic model for accessing multiple heterogeneous databases in a manufacturing environment. In *IMS '91 First International Workshop on Interoperability in Multidatabase Systems*, April. Kyoto, Japan.

Rosenthal, A. and Siegel, M. (1991). An architecture for practical metadata integration. In *Workshop on Information Technology Systems*, December. Cambridge, Massachussets.

Salton, G. (1989). *Automatic Text Processing.* Addison Wesley. Reading, Massachussets.

Sciore, E., Siegel, M. and Rosenthal, A. (1993). Using semantic values to facilitate interoperability among heterogeneous database systems. *ACM Transactions on Database Systems.*

Shafer, G. (1976). *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, N.J.

Sheth, A. and Gala, S. (1989). Attribute relationships: An impediment in automating schema integration. In *In the Workshop on Heterogeneous Database Systems* P. Scheuermann and C. Yu editors, Dec., Chicago, Il.

Sheth, A. and Larson, J. A. (1990). Federated database systems. *ACM Comput. Surv.*, 22(3).

Siegel, M. and Madnick, S. (1991). A metadata approach to resolving semantic conflicts. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, September.

Spaccapietra, S., Parent, C. and Dupont, Y. (1992). Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1:81–126.

Urban, S. D. (1991). A semantic framework for heterogeneous database environments. In *IMS '91 First International Workshop on Interoperability in Multidatabase Systems*, April. Kyoto, Japan.

Wang, R. and Madnick, S. (1989). The Inter-database instance identification problem in integrating autonomous systems. In *Proc. of the International Conference on Data Engineering*, pages 46-55.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *Computer*, 25(3), March, 1992.

Winograd, T. and Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design.* Norwood, N.J.: Ablex.

Yu, C. T., Jia, B., Sun, W. and Dao, S. (1991). Determining relationships among names in heterogeneous databases. *Sigmod Record Special Issue: Semantic Issues in Multidatabase Systems*, 20(4), December, 1991.