Representing and Using Interschema Knowledge in Cooperative Information Systems

Tiziana Catarci and Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria 113, I-00198 Roma, Italy
E-mail: [catarci@infokit,lenzerini@assi].ing.uniroma1.it

Abstract

Managing interschema knowledge is an essential task when dealing with cooperative information systems. We propose a logical approach to the problem of both expressing interschema knowledge, and reasoning about it. In particular, we set up a structured representation language for expressing semantic interdependencies between classes belonging to different database schemas, and present a method for reasoning over such interdependencies. The language and the associated reasoning technique makes it possible to build a logic-based module that can draw useful inferences whenever the need arises of both comparing and combining the knowledge represented in the various schemas. Notable examples of such inferences include checking the coherence of interschema knowledge, and providing integrated access to a cooperative information system.

This work has been supported by CNR, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 6, Linea di ricerca INFOKIT and Sottoprogetto 7, Linea di ricerca IBRIDI. An extended abstract of this paper appears in the *Proceedings of the International Conference on Intelligent and Cooperative Information Systems*, IEEE Computer Society Press, 1993.

1 Introduction

Understanding the semantics of data stored in different databases is an essential task when dealing with cooperative information systems. Recent work on interoperability (see for example [9, 19, 22]) points out that two individual information systems can interoperate on the basis of a mutual understanding of the information resources they provide. Obviously, in order to achieve this mutual understanding, several forms of interschema knowledge must be expressed and reasoned upon.

It is important to note that interschema knowledge is needed in a cooperative information system, independently of the architecture of the system. Following [12], we can distinguish between two basic approaches for accessing cooperative information systems. This first approach is based on a global schema describing the information in the individual databases, in such a way that users and applications are presented with the illusion of a single, centralized information system. The second approach avoids constructing the global schema, and relies on various tools to be used for sharing information by users and applications. It is evident that interschema knowledge is an essential element in both approaches: in the former, it provides the necessary information for building the global schema, while in the latter, it is used for understanding the content of different databases, so as to share relevant information.

In this paper, we investigate on the possibility of using logic for both expressing interschema knowledge, and reasoning about it. We assume that the individual information systems which are the components of the cooperative information system are defined in terms of the same data model, which is an class-oriented model. This assumption allows us to hide some technical problems related to the treatment of heterogeneous databases, that would deserve more details. However, since there exist formal translations between virtually all data models (see for example [16]) and the language we use here, the whole approach is still valid if we relax the above assumption.

The basic idea of our approach is to propose a logic-based language to express interdependencies between classes belonging to different schemas. Such interdependencies allow a designer of a cooperative information system to establish several relationships between both the intensional definition and the set of instances of classes represented in different schemas. For example, one can assert in our language that the concept represented by the class

`GraduateStudent` in the schema S_1 is the same as the concept represented by `SeniorStudent` in S_2. Such assertion implies a sort of intensional equivalence between the two classes, but does not imply that the set of instances of the former is always the same as the set of instances of the latter. On the other hand, one can assert that the set of instances of the class `Teacher` in the schema S_2 is always disjoint from the set of instances of the class `Tutor` in S_3, even if there exists a form of intensional inclusion between the latter and the former.

Once we have a set of assertions of the above mentioned kinds, we may benefit in several ways from the possibility of reasoning about them. One distinguishing feature of our work is to provide inference mechanisms that allow such reasoning to be carried out. Based on the formal semantics of the language, we can indeed devise suitable reasoning procedures that allow one to draw useful inferences on interschema knowledge. For example, one can check whether one class represented in the cooperative information system is incoherent, i.e. it has an empty extension in every state, or can deduce that the extension of a class A in the schema S_i is always a subset of the extension of the class B in S_j, so that accessing S_i is useless if we want to retrieve all the instances (stored in any of the information systems) of the concept represented by B.

Several recent papers in the literature share our general goal of representing and using interschema knowledge.

In [12], a method supporting the integrated acces to a set of information systems is proposed, based on the use of a global schema managed by the Cyc knwoledge representation system. One basic difference with our approach is the need for such a global schema that represents a sort of background knowledge for the whole set of information systems. As we shall see later, our approach is still valid if no common knowledge is available. Another difference with our work is that in [12] no distinction is made between intensional and extensional interdependencies between classes.

The problem of providing tools and techniques for the integrated access to a cooperative information system (either through a global schema, or through information sharing) is also addressed in [2, 14, 18, 17]. In [14, 18], an object-based common data model, similar to the one presented here, is adopted for expressing interschema knowledge. The main differences with our work are that only extensional interdependencies between classes are considered, and that reasoning does not play an important role in using interschema knowl-

edge. Both [2] and [17] advocate the use of a second-order language for representing several types of interschema dependencies. In particular, the main focus of [2] is to fully cope with representational hetereogeneity by providing suitable means for representing the constructs of different data models, whereas the main goal of [17] is to provide language features in order to deal with interoperable databases with schematic discrepancies. The problem addressed by these two last papers is more general in several aspects than the one studied here. On the other hand, compared with them, we are more concerned with representing intensional as well as extensional interdependencies between classes, and developing specialized reasoning techniques that allow for automatically exploiting interschema knowledge in various tasks.

Interschema knowledge is also the subject of [23], where the problem of maintaining consistency of related data in a multidatabase environment is studied. Finally, we mention several papers [15, 25, 28] that aim at developing methods and tools supporting the discovery of interschema dependencies. Since our techniques assume that a set of interschema assertions are given, this issue is somehow complementary to the one addressed in this paper.

The paper is organized as follows. In Section 2, we describe the basic characteristics of the class representation language we use for expressing the intensional level of individual information systems. In Section 3, we set up a logical language for expressing semantic interdependencies between classes belonging to different information systems. In Section 4, we address the problem of reasoning about interschema knowledge, and in Section 5, we illustrate how to make use of the interschema knowledge and the associated reasoning mechanisms when dealing with two important tasks, namely, checking interschema coherence, and providing integrated access to the cooperative information system. Finally, conclusions as well as a discussion on future development of our work are presented in Section 6.

2 Class representation language

In this section, we describe the formalism that we adopt for expressing the various schemas of a cooperative information system. In this formalism, the basic structure of a schema is expressed in terms of an entity-relationship-like notation [11]. Moreover, a logic-based language can be used for denoting complex class expressions, and for representing a rich variety of properties of

both entities and relationships. In the first subsection we describe the basic elements of our formalism, in the second subsection we present the language for denoting class expressions, and, finally, in the third subsection we discuss the expressive power of the formalism.

2.1 Basic elements of the representation language

As in most data models, in our model the universe of discourse is partitioned into two levels, called the instance level and the class level.

The basic elements of the *instance level* are *objects*, i.e. atomic things that can be identified through a unique symbol. Such a symbol plays the role of object identifier, like in many recent object-oriented data models [1]. As we shall see later, objects are grouped into classes, and the objects belonging to a class C are called the *instances* of C.

In order to reflect relevant associations, objects can be aggregated into *tuples*. The number of objects which are components of a tuple is called the arity of the tuple. Tuples of the same arity can be grouped into sets that are called relationships. The tuples belonging to a relationship R are called the *instances* of R.

The *class level*, which is specified in terms of a so-called *schema*, is constituted by an alphabet of class symbols, and by the specification of how the classes are related to each other. The *alphabet* is simply a list of symbols (or names) partitioned into entity, relationship, role, attribute, value, and domain symbols.

An *entity* is an abstraction for a class of objects (called its instances). The properties of an entity are modeled in terms of a set of attributes and a set of relationships with other entities.

A *relationship* is a class of tuples of objects of the same arity, described by a set of roles. Each relationship R has an associated arity, which is also the arity of the tuples that are instances of R. A *role* denotes a component of the relationship. A relationship has as many roles as its arity. For example, the relationship `Marriage`, of arity 2, has two associated roles, `Husband` and `Wife`. Every tuple that is an instance of a relationship has one component for each role of the relationship. Thus, every instance of `Marriage` has two components, one for the role `Husband` and one for the role `Wife`.

A *domain* is a set of values. We assume that a suitable set of constructs (such as enumeration, interval, basic domains like integer, string, etc.) in the

style of type definitions in conventional programming languages, are available for specifying the structures of domains. Also, we assume to have a collection of symbols representing values, i.e. elements of the domains. An *attribute* is a relation between an entity or a relationship, and a domain.

It is worth noting that in our model, attributes and relationships can be declared as functional (for example, the attribute `Age` of the entity `Person`). However, for the sake of simplicity, we do not consider this feature in this paper.

In the following, we use the term *class* as an abstraction for entity, relationship, attribute and domain.

2.2 Class expressions and assertions

Our model provides a rich variety of mechanisms for specifying meaningful properties of the classes of objects represented in the schema. The idea is that all the knowledge about the basic elements in the schema can be specified in terms of a set of assertions.

Syntactically, an *assertion* is a statement of the form

$$L_1 \ \dot{\leq} \ L_2$$

where L_1, L_2 are expressions in a suitable language, each one denoting a class. Informally, an assertion of the above form states that every instance of the class (denoted by the expression) L_1 is also an instance of the class L_2. As we shall se later, this assertional mechanism allow us to specify several interesting features of the classes.

Suppose we are given an alphabet of symbols B for a schema S, as specified in the previous subsection. We assume that B includes two special entity symbols, namely $\top$ and $\bot$, whose meaning will be clear later. We use the term *class expression* over B to denote any expression that is formed using the symbols in B according to the syntactic rules that we describe below. There are four kinds of class expressions, namely, entity, domain, relationship, and attribute expressions.

Let us discuss entity expressions first. The simplest type of *entity expression* is the so-called atomic expression, i.e. the one formed simply by the name of an entity. More complex entity expressions can be built by means of suitable constructors. In fact, the characteristics of the constructors we

use to form entity expressions resemble those of the so-called terminological or frame-based languages introduced in Artificial Intelligence (see for example [8, 13, 21]), whose connections with databases have been investigated in several ways [4, 6, 5, 7]. From a syntactic point of view, an entity expression over the alphabet B of the schema S is formed by means of the following rules (which implicitly define *domain expressions* too):

$$\begin{array}{rcl} C, F & \longrightarrow & E \mid \\ & & C \sqcap F \mid \\ & & C \sqcup F \mid \\ & & \neg C \mid \\ & & (\forall R[U].T_1 : C_1, \ldots, T_n : C_n) \mid \\ & & (\exists R[U].T_1 : C_1, \ldots, T_n : C_n) \mid \\ & & \forall A.D \mid \\ & & \exists A.D \\ D & \longrightarrow & V \mid \\ & & \{v_1, \ldots, v_n\} \end{array}$$

where the following metavariables are used:

- $E, R, T_1, \ldots, T_n, U, A, V$ denote symbols in the alphabet B. In particular, E denotes a name of an entity, R a name of a relationship, $T_1, \ldots, T_n, U$ denote names of roles, A a name of an attribute, and V a name of a domain,
- C and F denote entity expressions, and D denotes a domain expression over B,
- $v_1, \ldots, v_n$ denote symbols of values belonging to domains in B.

The intuitive meaning of entity expressions is as follows:

- $\top$ and $\bot$ represent the universal entity (the entity whose instances are all the objects in the schema) and the empty entity (the entity with no instance), respectively;
- E denotes the set of instances of the entity E;

- $C \sqcap F$ represents the set of instances of both C and F;
- $C \sqcup F$ represents the union of the set of instances of C and the set of instances of F;
- $\neg C$ represents the set of objects that are not instances of C;
- $(\forall R[U].T_1 : C_1, \ldots, T_n : C_n)$ represents the set of objects x such that for each tuple r of R with x as U-component, the T_i-component of r belongs to the extension of C_i;
- $(\exists R[U].T_1 : C_1, \ldots, T_n : C_n)$ represents the set of objects x such that there exists a tuple r of R with x as U-component and y as T_i-component, such that y belongs to the extension of C_i;
- $\forall A.D$ represents the set of objects that are associated with values in D by the attribute A;
- $\exists A.D$ represents the set of objects that are associated with at least one value in D by the attribute A.

We provide an example of entity expression. Suppose that `Student`, `GraduateStudent` and `GraduateCourse` are entity symbols, and `Enrolled` is a relationship symbol, with associated roles `s` and `c`. Then

$$\texttt{Student} \sqcap \neg \texttt{GraduateStudent} \sqcap \exists \texttt{Enrolled}[\texttt{s}].\texttt{c} : \texttt{GraduateCourse}$$

denotes the set students that are not graduate students and are enrolled in some graduate course.

Let us now turn our attention to relationship and attribute expressions. A *relationship expression* is simply an expression of the form

$$R[U_1, \ldots, U_n]$$

where R is a relationship name, and $\{U_1, \ldots, U_n\} = rol(R)$ are the roles associated to R. Finally an *attribute expression* is simply constitued by an attribute name A.

The formal semantics of class expressions is based on the notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \gamma^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a schema S with alphabet B consists of

- a nonempty set $\Delta^{\mathcal{I}}$ (the *universe* of $\mathcal{I}$, which is assumed to include all the values that are elements of the domains used in the schema),
- a function $\gamma^{\mathcal{I}}$ that maps every value to an element of $\Delta^{\mathcal{I}}$ in such a way that $\gamma^{\mathcal{I}}(a) \neq \gamma^{\mathcal{I}}(b)$ if $a \neq b$, and
- a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every entity expression to a subset of $\Delta^{\mathcal{I}}$, every attribute to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every relationship to a set of labeled tuples over $\Delta^{\mathcal{I}}$.

  In particular, if R is a relationship whose set of associated roles is $rol(R) = \{U_1, \ldots, U_m\}$, then $R^{\mathcal{I}}$ is a set of labeled tuples of the form $\langle U_1 : u_1, \ldots, U_m : u_m \rangle$, where $u_1, \ldots, u_m \in \Delta^{\mathcal{I}}$. Formally, a labeled tuple over $\Delta^{\mathcal{I}}$ that is an instance of R can be defined as a function from $rol(R)$ to $\Delta^{\mathcal{I}}$. In the following, if r is an instance of R, then we write $r[U_i]$ to denote the value associated with the U_i-component of the tuple r.

Moreover, in order for $\mathcal{I}$ to be an interpretation for S, the following equations must be satisfied:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
\{v_1, \ldots, v_n\}^{\mathcal{I}} &= \{\gamma^{\mathcal{I}}(v_1), \ldots, \gamma^{\mathcal{I}}(v_n)\} \\
(C \sqcap F)^{\mathcal{I}} &= C^{\mathcal{I}} \cap F^{\mathcal{I}} \\
(C \sqcup F)^{\mathcal{I}} &= C^{\mathcal{I}} \cup F^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid a \notin C^{\mathcal{I}}\} \\
(\forall R[U].T_1 : C_1, \ldots, T_n : C_n)^{\mathcal{I}} &= \{a \mid \forall r \in R^{\mathcal{I}}.(r[U] = a) \Rightarrow \\
&\quad (r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_n] \in C_n^{\mathcal{I}})\} \\
(\exists R[U].T_1 : C_1, \ldots, T_n : C_n)^{\mathcal{I}} &= \{a \mid \exists r \in R^{\mathcal{I}}.(r[U] = a) \wedge \\
&\quad r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_n] \in C_n^{\mathcal{I}}\} \\
(\forall A.V)^{\mathcal{I}} &= \{a \mid \forall (a, b) \in A^{\mathcal{I}}.b \in V^{\mathcal{I}}\} \\
(\exists A.V)^{\mathcal{I}} &= \{a \mid \exists (a, b) \in A^{\mathcal{I}}.\, b \in V^{\mathcal{I}}\} \\
(R[U_1, \ldots, U_n])^{\mathcal{I}} &= R^{\mathcal{I}}
\end{aligned}
$$

Note that the notion of interpretation for a schema S corresponds to the notion of database state for S. Notice also that, with the above definition

of interpretation, we have provided the semantics of class expressions in the schema S. In particular, the semantics of a class expression L in the schema S with respect to an interpretation $\mathcal{I}$ for S is simply given by $L^{\mathcal{I}}$.

We are now ready to formally specify the meaning of a set of assertions. As we said before, an *assertion* is a statement of the form

$$L_1 \ \dot{\leq} \ L_2$$

where L_1 and L_2 are class expressions of the same type (either entity, relationship, or attribute) over an alphabet B of the schema S. The semantics of $L_1 \ \dot{\leq} \ L_2$ is given by the following conditions:

- When L_1 and L_2 are entity or attribute expressions, the assertion is satisfied by an interpretation $\mathcal{I}$ for S just in case $L_1^{\mathcal{I}} \subseteq L_2^{\mathcal{I}}$.
- When $L_1 = R_1[U_1, \ldots, U_n]$ and $L_2 = R_2[T_1, \ldots, T_m]$ are relationship expressions, then it must be the case that $m = n$, and the assertion is satisfied by $\mathcal{I}$ if for every tuple $< U_1 : d_1, \ldots, U_n : d_n >$ in $R_1^{\mathcal{I}}$, the tuple $< T_1 : d_1, \ldots, T_n : d_n >$ is in $R_2^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is called a *model* of a set of assertions Σ if every assertion in the set is satisfied by $\mathcal{I}$.

To summarize, we conceive a schema S in our model as a pair $S =< B, \Sigma >$, where B is an alphabet of symbols and Σ is a set of (intraschema) assertions over B. The semantics of S is formally characterized by all the interpretations for S that satisfy every assertions in Σ, i.e. by all the *models of* S.

2.3 Expressiveness of the representation language

The representation formalism introduced in this paper allows for expressing several interesting properties of the classes of a schema. The goal of this subsection is to provide an account of such properties.

The first example of the use of assertions is concerned with the possibility of specifying the type of the various components of a relationship. In fact, if we want to impose that the U-component of every instance r of the relationship R is an instance of the entity C, we can use the assertion[1]:

$$(\exists R[U]) \ \dot{\leq} \ C.$$

[1]We use the notation $\exists R[U]$ as a shorthand for $\exists R[U].U : \top$.

For example, referring to a schema concerning the information about the organization of a conference, the two assertions

$$(\exists \texttt{WrittenBy}[\texttt{Writer}]) \dot{\leq} \texttt{Author}$$
$$(\exists \texttt{WrittenBy}[\texttt{Publication}]) \dot{\leq} \texttt{Paper}$$

can be used to specify that the relationship `WrittenBy` is typed with the entity `Author` (in the role `Writer`) and the entity `Paper` (in the role `Publication`).

In many recent data models, classes are organized in hierarchies, specified by means of a binary relation over entities, called ISA relation. The fact that the class L_1 is related to the class L_2 by means of the ISA relation in a schema S specifies that the set of instances of L_1 is a subset of the set of instances of L_2 in every interpretation (database state) of S. It is easy to see that the ISA relation is directly represented in our model in terms of assertions.

For example, to represent that authors, referees, and staff members are persons, the following assertions can be used:

$$\texttt{Author} \dot{\leq} \texttt{Person}$$
$$\texttt{Referee} \dot{\leq} \texttt{Person}$$
$$\texttt{StaffMember} \dot{\leq} \texttt{Person}.$$

Although in the above two examples the ISA relation is established between two entities, it is worth mentioning that in our formalism assertions can be stated on relationships too. This results in a much richer mechanisms for using the ISA relation in the schema, compared with most of the existing data models.

In contrast to most of the approaches proposed in the literature, our model allows negative information to be represented in the schema. One notable example of negative information is the one stating that the sets of instances of two entities are disjoint. For example, in the above schema one can represent the fact that the entities `Author` and `Referee` and are disjoint by means of the assertion

$$\texttt{Author} \dot{\leq} \neg\texttt{Referee}.$$

Another example of negative information is the one stating that a certain relationship is meaningless for an entity, i.e. that the instances of the entity C cannot participate in the relationship R in role U. This is expressed through the assertion

$$C \;\dot{\leq}\; \neg\exists R[U].$$

Several kinds of indefinite information can also be expressed by means of assertions. Perhaps, the most important kind of indefinite assertion is related to the use of disjunction. For example, one can represent that every person is a male or a female by means of the assertion:

$$\texttt{Person} \;\dot{\leq}\; (\texttt{Male} \sqcup \texttt{Female}).$$

As another example, suppose that we want to state that no persons other than authors, referees and staff members are of interest to the application. This can be represented by

$$\texttt{Person} \;\dot{\leq}\; (\texttt{Author} \sqcup \texttt{Referee} \sqcup \texttt{StaffMember}).$$

Finally, our model allows one to specify both necessary and sufficient conditions for an object to be an instance of a class. The combined use of necessary and sufficient conditions enables the designer to provide a sort of *definition* of entities, which is closely related to the notion of view in databases. For example, the assertions

$$\begin{aligned} \texttt{AcmConference} \;&\doteq\; (\texttt{Conference} \sqcap (\exists \texttt{SponsoredBy}[\texttt{Event}].\texttt{Sponsor} : \\ &\qquad \texttt{AcmOrganization}) \\ \texttt{ItalianPaper} \;&\doteq\; \texttt{Paper} \sqcap \forall \texttt{WrittenBy}[\texttt{Publication}].\texttt{Author} : \texttt{Italian} \end{aligned}$$

can be used in order to define the entities `AcmConference` and `ItalianPaper`.

The above observations show that several interesting modeling constructs are available in our model, all based on the notion of class, which is generally recognized as the central notion of most of the existing representation models [20]. By carrying out an analysis of the main constructs available in different models, it is possible to show that the most of the data models proposed in the literature are actually subsumed by our model. This is particularly important in our context, where we assume that, in case the cooperative information system adopt different data models, a translation step is carried out for expressing the various schema in terms of our class representation language.

3 Specification of interschema knowledge

In this section, we describe our method for specifying interschema knowledge in terms of interdependencies between classes belonging to different schemas. We first describe the form of interschema assertions and their intuitive meaning, and then introduce the notion of cooperative information system in our context, whose formal semantics determines the semantics of interschema assertions.

3.1 Interschema assertions

Suppose that the cooperative information system is constituited by n individual information systems, called component information systems, whose schemas are $S_1, \ldots, S_n$, with alphabets $B_1, \ldots, B_n$. We assume that all the schema are expressed in the representation formalism presented in Section 2.

Since we want to keep separated the symbols of the various schemas, we also assume that for every $i \in \{1..n\}$, the elements of B_i are subscripted with i. Thus, for example, $\texttt{Teacher}_2$ denotes a class of schema 2, named Teacher. We use the term S_i-class expression to denote any class expression over the alphabet B_i of the schema S_i. For example, if $\texttt{Student}_1$ and $\texttt{Woman}_1$ are entity symbols in B_1, then $(\texttt{Student}_1 \sqcap \texttt{Woman}_1)$ is an S_1-entity expression. Notice that the symbols used in an S_i-class expression must belong to the alphabet B_i of S_i.

Let $S_0 =< B_0, \Sigma_0 >$ be a further schema, called the *common knowledge schema* of the cooperative information system (symbols in B_0 are subscripted with 0). Intuitively, S_0 represents the general properties of the classes that should be considered common knowledge in the cooperative information system of interest. Obviously, in those applications where such a knowledge is not available, the set of assertions in Σ_0 will be empty. In this sense, our framework differs from those approaches where all interschema properties are established through the use of background knowledge (see for example [12]) to which the various component information systems must conform. On the other hand, if the common knowledge schema is nonempty, then it is reasonable to require all the other schemas to be coherent with it. We will come back to this issue in Section 5.

Please, note that the notion of common knowledge schema is also different from the one of global schema in the context of schema integration [3], in

that the former is intended to represent an integrated view of all the classes of interest in the collection of information systems, while the latter is used to specify the knowledge that is independent of the views of the world as reflected in the individual schemas. A consequence of this is that, as we said before, the common knowledge schema may be empty, whereas the global schema is always non empty.

Interschema knowledge in our approach is specified in terms of a set of so-called *interschema assertions*. There are four kinds of assertions, whose forms are:

$$\begin{array}{lll} L_1 & \doteq_{int} & L_2 \\ L_1 & \dot{\leq}_{int} & L_2 \\ L_1 & \doteq_{ext} & L_2 \\ L_1 & \dot{\leq}_{ext} & L_2 \end{array}$$

where in every assertion, L_1 represents an S_i-class expression, and L_2 represents an S_j-class expression, in such a way that L_1 and L_2 are class expressions of the same types (either entity, domain, relationship, or attribute expressions), and $i \neq j$. Moreover, if L_1 and L_2 are relationships expressions, then there is the constraint that the set of roles appearing in L_1 has the same cardinality as the set of roles appearing in L_2. In other words, an interschema assertion represents a well-typed interdependency between two class expressions belonging to two different schema.

We now discuss the intuitive meaning of the four kinds of interschema assertions that we have introduced, and then present the formal semantics. For the sake of simplicity, in the observations on the intuitive meaning of interschema assertions, we only refer to assertions on entities.

The first assertion states that the entity expression L_1 is intensionally equivalent to L_2. Intuitively, this means that, if S_i and S_j referred to a unique set of objects in the real world, then the extension of L_1 would be the same as the extension of L_2. Therefore, the above assertion is intended to state that, although the extension of L_1 may be different from the extension of L_2, the concept represented by L_1 is in fact the same as the concept represented by L_2. As a simple example, the S_1-entity $\texttt{UndergraduateStudent}_1$ can be declared intensionally equivalent to the S_1-entity $(\texttt{Student}_2 \sqcap \neg\, \texttt{GraduateStudent}_2)$, to reflect that, even if the instances of the two expressions may be different in the various states of the cooperative information system, the concept of

`UndergraduateStudent` in the schema S_1 is fully captured by the above entity expression in the schema S_2.

The second assertion states that the entity expression L_1 (say of schema S_i) is intensionally less general than the entity expression L_2 (of schema S_j). This means that there is a sort of containment between L_1 and L_2, and this containment is conceptual, not necessarily being reflected at the instance level. In other words, the above intensional relationship is intended to state that, if S_i and S_j referred to a unique set of objects in the real world, then the extension of L_1 would be a subset of L_2. For example, $\texttt{Tutor}_2$ may be declared intensionally less general than $\texttt{Teacher}_3$, if the concept of tutor as represented in the schema S_2 is subsumed by the concept of teacher in the schema S_3.

The third assertion states that L_1 and L_2 are always extensionally equivalent. In this case we are asserting that in every state of the cooperative information system, the set of objects that are instances of L_1 in S_i is the same as the set of objects that are instances of L_2 in S_j.

Finally, the fourth assertion states that the extension of L_1 is always a subset of the extension of L_2. For example, if the entity $\texttt{Student}_1$ refers to the students of a University Department, and the schema S_2 refers to the whole University, then we may assert that $\texttt{Student}_1$ is extensionally less general than $\texttt{Student}_2$.

3.2 Semantics of interschema assertions

In order to assign the formal semantics to assertions, we introduce the notion of cooperative information system, and the notion of interpretation for it.

A *cooperative information system* is intended to precisely characterize all the knowledge represented in the cooperative information system. At the class level, a cooperative information system $G =< S_0, S_1, \ldots, S_n, \Sigma >$ is constituted by

- the collection of the n schemas $S_1, \ldots, S_n$ of the various information systems,
- a common knowledge schema S_0 (which may be empty),
- a set Σ of interschema assertions of the forms presented above.

An *interpretation* $\mathcal{I}$ for $G =< S_0, S_1, \ldots, S_n, \Sigma >$ is built up starting from n interpretations $\mathcal{I}_1, \ldots, \mathcal{I}_n$ for $S_1, \ldots, S_n$, respectively. In particular, $\mathcal{I}$ is defined as a triple $(\Delta^{\mathcal{I}}, \gamma^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that

- $\Delta^{\mathcal{I}} = \Delta_0^{\mathcal{I}}$ is the union of the various $\Delta_i^{\mathcal{I}}$, for $i \in \{1, \ldots, n\}$,
- $\gamma^{\mathcal{I}}$ and $\cdot^{\mathcal{I}}$ are extensions of the union of the various $\gamma_i^{\mathcal{I}}$ and $\cdot_i^{\mathcal{I}}$ ($i \in \{1..n\}$), respectively, in such a way that every symbol in S_0 is given an extension by $\mathcal{I}$.

We now state the conditions for an interpretation to satisfy an inter-schema assertion. Let $\mathcal{I}$ be an interpretation for $G =< S_0, S_1, \ldots, S_n, \Sigma >$. We assume that in the assertion $C_1 \doteq_\beta C_2$, where β is either *int* or *ext*, C_1 is an S_i-class expression and C_2 is an S_j-class expression.

- The assertion
$$C \dot{\leq}_{int} F$$
where C and F are entity expressions, is satisfied by $\mathcal{I}$ if for every $a \in \Delta_i^{\mathcal{I}} \cap \Delta_j^{\mathcal{I}}$, $a \in C^{\mathcal{I}}$ implies $a \in F^{\mathcal{I}}$.

- The assertion
$$R_1[U_1, \ldots, U_n] \dot{\leq}_{int} R_2[T_1, \ldots, T_n]$$
where R_1 and R_2 are relationship expressions, is satisfied by $\mathcal{I}$ if for every $u_1, \ldots, u_n \in \Delta_i^{\mathcal{I}} \cap \Delta_j^{\mathcal{I}}$, $< U_1 : u_1, \ldots, U_n : u_n > \in R_1^{\mathcal{I}}$ implies $< T_1 : u_1, \ldots, T_n : u_n > \in R_2^{\mathcal{I}}$.

- The assertion
$$L_1 \doteq_{int} L_2$$
where L_1 and L_2 are class expressions, is satisfied by $\mathcal{I}$ if both $L_1 \dot{\leq}_{int} L_2$ and $L_2 \dot{\leq}_{int} L_1$ are satisfied by $\mathcal{I}$.

- The assertion
$$C \dot{\leq}_{ext} F$$
where C and F are entity expressions, is satisfied by $\mathcal{I}$ if for every $a \in \Delta^{\mathcal{I}}$, $a \in C^{\mathcal{I}}$ implies $a \in F^{\mathcal{I}}$.

- The assertion

$$R_1[U_1, \ldots, U_n] \quad \doteq_{ext} \quad R_2[T_1, \ldots, T_n]$$

  where R_1 and R_2 are relationships, is satisfied by $\mathcal{I}$ if for every $u_1, \ldots, u_n \in \Delta^{\mathcal{I}}$, $< U_1 : u_1, \ldots, U_n : u_n > \in R_1^{\mathcal{I}}$ implies $< T_1 : u_1, \ldots, T_n : u_n > \in R_2^{\mathcal{I}}$.

- The assertion

$$L_1 \quad \doteq_{ext} \quad L_2$$

  where L_1 and L_2 are class expressions, is satisfied by $\mathcal{I}$ if both $L_1 \ \dot{\leq}_{ext} \ L_2$ and $L_2 \ \dot{\leq}_{ext} \ L_1$ are satisfied by $\mathcal{I}$.

Analogous definitions hold for the semantics of assertions on attribute expressions.

An interpretation $\mathcal{I}$ for a cooperative information system $G =< S_0, S_1, \ldots, S_n, \Sigma >$ is called a *model* of G if every assertion in Σ is satisfied by $\mathcal{I}$.

If σ is an (intraschema or interschema) assertion, we say that σ is a *logical consequence* of G (written $G \models \sigma$) if σ is satisfied by every model of G.

4 Reasoning on interschema assertions

Most of the reasoning we want to perform on a cooperative information system G can be reduced to the problem of checking whether $G \models \sigma$, where σ is an assertion. In this section, we describe a technique that can be used to devise a solution to this problem. We then show how to use the technique in order to build a so-called interschema knowledge network, which is a sort of semantic network representing the relevant knowledge about the interdependencies holding between classes of different schemas.

4.1 Basic reasoning technique

Our reasoning technique is based on the idea of translating G into a set K^G of assertions of the form

$$L_1 \ \dot{\leq}_{ext} \ L_2$$

and then taking advantage of a correspondence between sets of assertions in our language and formulae of a variant of propositional dynamic logic. In

particular, every element of K^G is an assertion over the alphabet B which is the union of the alphabets of the schemas $S_0, S_1, \ldots, S_n$.

The translation of G into K^G is done as follows (where L_1 is an S_i-class expression and L_2 is an S_j-class expression):

- first of all, we include in K^G all intraschema assertions of each of the schemas $S_0, S_1, \ldots, S_n$,
- we include in K^G the assertions

$$\begin{array}{rcl} \top_0 & \dot{\leq}_{ext} & \top_1 \sqcup \cdots \sqcup \top_n \\ \top_1 \sqcup \cdots \sqcup \top_n & \dot{\leq}_{ext} & \top_0 \end{array}$$

  in order to reflect the semantic constraint that $\Delta_0^{\mathcal{I}}$ is the union of the various $\Delta_i^{\mathcal{I}}$, for $i \in \{1, \ldots, n\}$),
- for each assertion σ of the form $L_1 \dot{\leq}_{ext} L_2$ in Σ, we include σ in K^G,
- for each assertion of the form $L_1 \doteq_{ext} L_2$ in Σ, we include in K^G the assertions

$$\begin{array}{rcl} L_1 & \dot{\leq}_{ext} & L_2 \\ L_2 & \dot{\leq}_{ext} & L_1 \end{array}$$

- for each assertion of the form $L_1 \doteq_{int} L_2$ in Σ, we include in K^G the assertions

$$\begin{array}{rcl} L_1 \sqcap \top_j & \dot{\leq}_{ext} & L_2 \\ L_2 \sqcap \top_i & \dot{\leq}_{ext} & L_1 \end{array}$$

- for each assertion of the form $L_1 \dot{\leq}_{int} L_2$ in Σ, we include the assertion

$$L_1 \sqcap \top_j \ \dot{\leq}_{ext} \ L_2$$

It is possible to show that, for any $G =< S_0, S_1, \ldots, S_n, \Sigma >$, K^G correctly captures the relevant semantics of the assertions in Σ.

The problem is now how to reason on K^G, i.e. how to check whether $K^G \models \sigma$, where σ is an assertion. To this end, we establish an interesting

correspondence between the set of assertion in K^G and a formula in a special propositional dynamic logic. A similar correspondence was noticed by Schild [24] between terminological languages and modal logics of programs. This method allows us to prove that reasoning in our language is decidable.

Propositional dynamic logic (PDL) is an extension of propositional logic in order to reason about program schemes. The language of PDL includes special operators for forming programs and special modal operators for stating what is true in the states resulting from the execution of the programs. From a semantic point of view, formulae of PDL are interpreted in a set of states, so that programs correspond to transitions over such states, and formulae represent conditions that can be true or false in the variuos states. We refer, in particular, to Converse-Deterministic-PDL (CDPDL), that is an extension of the basic PDL to deal with backward computations (inverse of the transition function corresponding to a program).

Roughly speaking, we associate with K^G a CDPDL formula $\phi(K^G)$ in such a way that models of K^G corresponds to models of $\phi(K^G)$, and viceversa. The correspondence between models is based on a one-to-one mapping between objects in the interpretations of K^G and states in the interpretations of $\phi(K^G)$, and between roles (recall that roles connect entities to relationships) and programs in $\phi(K^G)$.

With the above approach, the logical implication problem for K^G is reduced to the analogous problem for $\phi(K^G)$. Since the size of $\phi(K^G)$ is polynomially related to the size of K^G, and reasoning over CDPDL formulae has an exponential time upper bound [27], it follows that reasoning over K^G is decidable, with exponential time as worst case complexity. More details about the method can be found in [10].

4.2 Interschema knowledge network

While the technique we have described is the basis for reasoning about a cooperative information system G, it is often useful to embedd the knowledge represented in G into a compact structure, reflecting all the relevant interdependencies holding between classes of different schemas. In particular, in our approach, the relevant interschema knowledge can be expressed in terms of a so-called *interschema knowledge network*, which is defined as a directed graph whose nodes represent classes, and arcs represent extensional containment between classes.

The construction of the interschema knowledge network N^G associated with a cooperative information system G is again based on the above described reasoning technique, and proceeds as follows:

1. select from the schemas $S_0, S_1, \ldots, S_n$ the classes that are to be associated with the nodes of N^G,

2. for every pair of selected classes L_1, L_2, draw an arc in N^G from the node corresponding to L_1 to the node corresponding to L_2, if $G \models L_1 \stackrel{\cdot}{\leq}_{ext} L_2$,

3. compute the transitive reduction of N^G, i.e. iteratively remove from N^G all the arcs from node n_1 to node n_2 such that there is a node n_3 with arcs $< n_1, n_3 >$ and $< n_3, n_2 >$ in N^G,

4. for every cycle γ in N^G, merge all the nodes belonging to γ into a single node.

By comparing the interschema knowledge network N^G with G itself, we can point out three main characteristics of N^G.

First, N^G refers to a selected subset of (not necessarily all) the classes in the various schemas constituting G. There are several possibility for such a selection. For example, we may select only atomic entities from the various schemas, or we may include all atomic entities plus some important complex entities, or we may want to represent in N^G all the entities denoted by some expression in the interschema assertions. Note that the choice of which entities are represented in N^G determines both the complexity and the completeness of the information carried by N^G itself.

Second, while G is a flat collection of interschema assertions, N^G reflects the structure of interdependencies between classes, by making explicit all the extensional inclusions between selected classes that are implied by G. Note that such a structure results in a partial order between the selected classes.

Finally, the arcs of N^G reflect some important inferences that can be drawn from G. In particular, the existence of a path in N^G from two nodes n_1 and n_2 implies that all instances of the class represented by n_1 is also an instance of the class represented by n_2. Moreover, the mutual extensional equivalence of a set of selected classes is made explicit in N^G by merging the corresponding nodes.

The next section is devoted to a discussion on how to use the interschema knowledge network N^G in order to support several types of tasks to be accomplished on the cooperative information system G.

5 Using interschema knowledge

In this section, we show how we can make use of interschema assertions and the associated reasoning mechanisms in order to draw interesting inferences on the knowledge represented in a cooperative information system. We discuss this issue by distinguish two main aspects where reasoning can play an important role, namely interschema consistency, and integrated access to the cooperative information system.

5.1 Interschema consistency

Besides providing information on the correspondence between classes in different schemas, interschema assertions actually constitute a declarative specification of several consistency requirements over different databases. For example, the interschema assertion

$$\texttt{GraduateStudent}_2 \ \dot{\leq}_{ext} \ \texttt{Student}_1$$

specifies that any update operation inserting new instances into the class `GraduateStudent` of schema S_2 should ensure that the same instance is also in the class `Student` of schema S_1. Obviously, if the interschema knowledge is itself incoherent, then no state of the cooperative information system may exists satisfying all the interschema assertions. Therefore checking coherence is one of the basic activities for verifying the correctness of the cooperative information system. In our framework, coherence verification corresponds to checking G for satisfiability (G is said to be satisfiable if it admits at least one model), and this task can be directly done by exploiting the reasoning technique described in the previous section.

Even if the set of interschema assertions is globally satisfiable, it may be the case that the cooperative information system suffers from other types of incoherence. Let us call a class L *incoherent* in G if the extension of L is invariably empty in all the models of G. Class coherence can be checked by using the interschema knowledge network. Indeed, it is easy to see that the

fact that a class L_i to be incoherent in G is reflected in the existence of a path in N^G from the node corresponding to L to the node associated with $\bot_0$.

A further aspect related to the consistency of interschema knowledge is concerned with schema integration. We argue that the whole framework presented in this paper provides a *formal* setting for schema integration, an aspect that is often neglected in the existing integration methodologies. If we want to integrate two or more schemas belonging to the cooperative information system, we can benefit from the knowledge expressed in N^G, in all the phases of the integration process (see [3, 26]). In particular, the activity of conflict detection can be fully automatized in our approach. Also, N^G allows one to single out several forms of redundancies in the integrated schemas, and makes it explicit which are the links to be included in the integrated schema in order to reflect meaningful relationships between classes coming from different schemas.

Notice that the issue of interschema consistency checking is not only relevant when a global view of the cooperative information system is to be constructed. In fact, checking the coherence of interschema knowledge is required every time information is shared among two or more databases. For this reason, this issue is also important in those approaches where building a global schema is avoided.

5.2 Integrated access

We use the term integrated access to mean any querying operation that may require accessing different component information systems. As pointed out in [17], typical needs requiring integrated access include: to pose queries with the same intention to each component informations system, to pose queries spanning over several component information systems, to be provided with a unified view of the cooperative information system.

Our goal in this section is to show that interschema assertions and its associated reasoning mechanisms can play an important role in supporting integrated access, in particular by allowing several forms of intensional reasoning to be performed over query expressions.

We assume that queries are complex expressions whose atomic components are class expressions in the language described in Section 2. Such atomic components, called *query clss expressions*, are the elements that the

system can reason upon. We also assume that all symbols in $B_0, B_1, \cdots, B_n$ may appear in query class expressions, so that users and applications can refer both to selected schemas, when classes of objects of a specific informations system are requested, and to the common knowledge schema, when general classes, independently of the location of the corresponding instances, are needed. For example, a query Q may refer to

$$\texttt{Student}_0 \sqcap \exists \texttt{Enrolled}_0[\texttt{s}].\texttt{c} : \texttt{GraduateCourse}_1$$

denoting the classes of objects of the common knowledge schema that are instances of `Student` and are enrolled in at least one course that is an instance of the class $\texttt{GraduateCourse}_1$ of schema S_1. Notice that the formal semantics of a query class expression easily derives from the semantics of the cooperative information system G.

All intensional reasoning performed on a query class expression Q is based on first classifying Q over the interschema knowledge network N^G associated with a cooperative information system G, and then analysing the result of such classification in order to deduce relevant information for the query answering strategy.

Classifying Q over N^G means

1. introducing a new node D corresponding to Q in the network,
2. drawing an arc from D to E for each node E such that $K^G \models (D \ \dot{\leq}_{ext} \ E)$,
3. drawing an arc from F to D for each node F such that $K^G \models (F \ \dot{\leq}_{ext} \ D)$,
4. and, finally, computing the transitive reduction of the network, and merging nodes corresponding to equivalent, thus obtaining the network N^G_Q.

The resulting network N^G_Q carries out several interesting information about Q. We analyze three interesting cases where the system can take advantage of such information.

1. The node D is placed under the node associated with $\bot_0$, meaning that the query represents class that is incoherent in G. It is evident that such an information allow the system to avoid accessing the cooperative information system, thus saving processing time.

2. The node D is merged with a node Z associated with a class C belonging to a schema S_i, with $i \neq 0$. In this case, the query class Q is extensionally equivalent to C, and the answer to the query can be obtained simply by accessing the component information system whose schema is S_i.

3. If none of the above two cases occur, then let $F_1, \ldots, F_p$ be the immediate successors of D such that each $i \in \{1, \ldots, p\}$ is different from 0, and let $H_1, \ldots, H_q$ be the immediate predecessors of D such that each $i \in \{1, \ldots, q\}$ is different from 0. The basic observation is that possible redundancies between $F_1, \ldots, F_p$ $(H_1, \ldots, H_q)$ and other classes in the cooperative information system are made explicit by the structure of N_Q^G. For example, it is clear that accessing a class C_j which is a predecessor of H_k for some $K \in \{1, \ldots, q\}$, is useless for retrieving the instances of Q.

   We illustrate this case by means of an example. Suppose that the schema S_1 refers to the University U and the schema S_2 refers to the Computer Science Department of U, and consider the following inter-schema assertions:

$$
\begin{aligned}
\texttt{Student}_2 &\ \dot{\leq}_{ext}\ \texttt{Student}_1 \\
\texttt{AdvancedCSC}_1 &\ \doteq_{ext}\ \texttt{AdvancedCSC}_2 \\
\texttt{CSCourse}_1 &\ \doteq_{ext}\ \texttt{CSCourse}_2 \\
\texttt{Enrolled}_1[\texttt{s},\texttt{c}] &\ \doteq_{int}\ \texttt{Enrolled}_2[\texttt{s},\texttt{c}] \\
\texttt{Enrolled}_2[\texttt{s},\texttt{c}] &\ \dot{\leq}_{ext}\ \texttt{Enrolled}_1[\texttt{s},\texttt{c}] \\
\texttt{Tutoring}_0[\texttt{t},\texttt{c}] &\ \doteq_{int}\ \texttt{Tutoring}_1[\texttt{t},\texttt{c}] \\
\texttt{Tutoring}_0[\texttt{t},\texttt{c}] &\ \doteq_{int}\ \texttt{Tutoring}_2[\texttt{t},\texttt{c}] \\
\exists \texttt{Tutoring}_0[\texttt{t}].\texttt{c} : \texttt{CSCourse}_0 &\ \doteq_{int}\ \texttt{Student}_2 \\
\exists \texttt{Enrolled}_1[\texttt{s}].\texttt{c} : \texttt{AdvancedCSC}_1 &\ \dot{\leq}_{ext}\ \texttt{Student}_2.
\end{aligned}
$$

   The last two assertions state that the class of tutors of advanced Computer Science courses is intensionally less general than the class of students of interest for the Computer Science Department, and that the instances of the class `Student` of the schema S_1 that are enrolled in some advanced Computer Science course are also instances of the class

Student of the schema S_2. Now, if we classify the query class expression Q

$$\begin{aligned} \texttt{Student}_1 \ &\sqcap \ (\exists \texttt{Enrolled}_2[\texttt{s}].\texttt{c} : \texttt{AdvancedCSC}_2) \\ &\sqcap \ (\exists \texttt{Tutoring}_0[\texttt{t}].\texttt{c} : \texttt{CSCourse}_0) \end{aligned}$$

over the interschema knowledge network, it turns out that Q is a subset of $\texttt{Student}_2$ and therefore, although the query refers to symbols in S_0, S_1 and S_2, it is sufficient to access only S_2 in order to compute the answer.

All the above observations show that, based on the fact that the various nodes (except for those associated with S_0-classes) correspond to classes whose instances are located in a particular component information system, the system can profitly take advantage of N_Q^G in order to decide which are the minimal collection of component information systems to be accessed in order to retrieve the instances required by the query.

Other types of reasoning over N_Q^G, such as deciding if accessing a given set of component information systems is sufficient for answering Q, or reformulating an S_0-query class expression in terms of the symbols of a given alphabet B_i (with $i \neq 0$), can also be carried out by exploiting the information provided by N_Q^G.

6 Conclusions

We have presented a formal framework for representing interschema knowledge in cooperative information systems. Interschema knowledge is specified in terms of a special language, allowing for expressing several forms of semantic interdependencies between the classes represented in different schemas. One distinguishing feature of our work is to provide inference mechanisms that allow one to reason on such assertions. We have discussed the use of the inference procedures in various tasks related to the use of cooperative information systems.

In this paper, we have assumed that all the schemas of interest are expressed in the same representation model, which is a class-oriented model based on first-order logic. However, as we said in the introduction, the whole approach can be extended to the case of multiple models.

We believe that the framework presented in this paper constitute a sound basis for studying the various issues related to interschema knowledge representation and reasoning. At the same time, the work presented in this paper makes it clear that there are several aspects that deserve more investigation. We point out here two important aspects.

First of all, the issue of efficiency of the reasoning procedure must be addressed. As we said before, the worst case computational complexity of such procedure is exponential. It is worth analyzing in more detail its behaviour in practical cases, and devising several optimizations. Notice, however, that the high complexity is concerned with the task of checking whether G logically imply an assertion σ (which is necessary for building N^G), whereas, after constructing N^G, we can make reasoning much more efficient. Also, although in this paper we have focused on complete (in the logical sense) reasoning procedures, we observe that many interesting kinds of reasoning needed in the framework of cooperative information systems can be carried out by incomplete, but more efficient, inference algorithms approximating the complete answer. We believe that this direction is a promising one for effectively using in practical settings the logic underlying our approach.

Another challenging problem to consider is related to the need of going beyond the realm of first order logic when expressing interschema knowledge. In particular, it would be useful for a designer of a cooperative information system to express several forms of default assumptions on interschema assertions. A simple and important example of such assumptions is the one stating that, by default, two classes that are not explicitely linked in the interschema knowledge network are in fact disjoint. This is a sort of closed world assumption regarding the schema level of the cooperative information system, that generalize the usual closed world assumption in traditional databases. More generally, it would be interesting to study to what extent the existing approaches to nonmonotonic reasoning developed in Artificial Intelligence are well suited for dealing with default knowledge in cooperative information systems.

References

[1] S. Abiteboul, P. C. Kanellakis. "Object identity as a query language." *Proceedings of the 1989 ACM SIGMOD International Conference on Man-*

agement of Data, 1989.

[2] T. Barsalou, D. Gangopadhyay. "M(DM): An open framework for interoperation of multimodel multidatabase systems." *Proceedings of the 8th IEEE Data Engineering Conference,* 1992.

[3] C. Batini, M. Lenzerini, S. B. Navathe. "A comparative analysis of methodologies for database schema integration." *ACM Computing Surveys,* **18:4**, 1986.

[4] H. W. Beck, S. K. Gala, S. B. Navathe. "Classification as a query processing technique in the CANDIDE semantic data model." *Proceedings of the 5th IEEE Data Engineering Conference,* 1989.

[5] D. Beneventano, S. Bergamaschi. "Subsumption for complex object data models." *Proceedings of the 4the International Conference on Database Theory,* 1992.

[6] S. Bergamaschi, C. Sartori. "On taxonomic reasoning in conceptual design." *ACM Transactions on Database Systems,* **17:3**, 1992.

[7] A. Borgida, R. J. Brachman, D. L. McGuinness, L. A. Resnick. "CLASSIC: A structural data model for objects." *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data,* 1989.

[8] R. J. Brachman, J. Schmolze. "An overview of the KL-ONE knowledge representation system." *Cognitive Science,* **9:2**, 1985.

[9] M. L. Broadie, S. Ceri. "On intelligent and cooperative information systems: a workshop summary." *International Journal of Intelligent and Cooperative Information Systems,* **1:1**, 1992.

[10] T. Catarci, M. Lenzerini. "Reasoning about entities and relationships." Submitted for publication, 1993.

[11] P. P. Chen. "The Entity-Relationship model – Toward a unified view of data." *ACM Transactions on Database Systems,* **1:1**, 1976.

[12] C. Collet, M. N. Huhns, W. Shen. "Resource integration using a large knowledge base in Carnot." *IEEE Computer,* **24:12**, 1991.

[13] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "The complexity of concept languages." *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, 1991.

[14] D. Fang, J. Hammer, D. McLeod. "The identification and resolution of semantic heterogeneity in multidatabase systems." *Proceedings of the 1st IEEE International Workshop on Interoperability in Database Systems*, 1991.

[15] J. Geller, Y. Perl, E. Neuhold. "Structural schema integration in heterogeneous multi-database systems using the dual model." *Proceedings of the 1st IEEE International Workshop on Interoperability in Database Systems*, 1991.

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

[17] R. Krishnamurthy, W. Litwin, W. Kent. "Language features for interoperability of databases with schematic discrepancies." *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, 1991.

[18] Q. Li, D. McLeod. "An object-oriented approach to federated databases." *Proceedings of the 1st IEEE International Workshop on Interoperability in Database Systems*, 1991.

[19] W. Litwin, L. Mark, N. Roussopoulos. "Interoperability of multiple autonomous databases." *ACM Computing Surveys*, **22:3**, 1990.

[20] R. Motschnig-Pitrik, J. Mylopolous. "Classes and instances." *International Journal of Intelligent and Cooperative Information Systems*, **1:1**, 1992.

[21] B. Nebel. *Reasoning and revisions in hybrid representation systems.* Lecture Notes in Artificial Intelligence, Springer-Verlag, 1990.

[22] M. P. Papazoglou, S. C. Laufmann, T. K. Sellis. "An organizational framework for cooperating information systems." *International Journal of Intelligent and Cooperative Information Systems*, **1:1**, 1992.

[23] M. Rusinkiewicz, A. Sheth, G. Karabatis. "Specifying interdatabase dependencies in a multidatabase environment." *IEEE Computer,* **24:12**, 1991.

[24] K. Schild. "A correspondence theory for terminological logics: preliminary report." *Proceeedings of the International Joint Conference on Artificial Intelligence,* 1991.

[25] A. Sheth, V. Kashyap. "So far (schematically), so near (semantically)." *Proceedings of the IFIP DS-5 Conference on Semantics of Interoperable Database Systems,* Elsevier Publisher, 1992.

[26] S. Spaccapietra, C. Parent, Y. Dupont. "Model independent assertions for integration of heterogeneous schemas." *VLDB Journal,* **1**, 1992.

[27] M. Y. Vardi, P. Wolper. "Automata-theoretic techniques for modal logics of programs." *Journal of Computer and System Science,* **32**, 1986.

[28] C. Yu, W. Sun, S. Dao, D. Keirsey. "Determining relationships among attributes for interoperability of multi-database systems." *Proceedings of the 1st IEEE International Workshop on Interoperability in Database Systems,* 1991.