

AN OBJECT-ORIENTED RULE-BASED APPROACH TO DATA MODEL AND SCHEMA TRANSLATION*

S. Y. W. Su S. C. Fang H. Lam

Database Systems Research and Development Center, CSE#470
Department of Computer and Information Sciences
Department of Electrical Engineering
University of Florida, Gainesville, FL 32611
Email: su@pacer.cis.ufl.edu, {sf, hlam}@reef.cis.ufl.edu

ABSTRACT

To achieve data sharing among heterogeneous database management systems, one essential step is the conversion of the schemata defined in the diverse data models used by these systems. A semantic preserving translation of different modeling constructs and constraints is necessary to ensure that the semantics of applications remain intact after the translation. In this paper, we present an object-oriented extensible core model (ORECOM) which is used as a neutral, intermediate model through which diverse modeling constructs and constraints are translated. The model provides very general and primitive structural constructs in the forms of objects and binary object associations for representing the structural properties of various high-level data models. It also provides behavioral constructs in the forms of operation specifications and semantic rules with triggers for capturing the semantic constraints of these high-level models. Different from the intermediate models used in the existing heterogeneous database management systems, ORECOM is developed for supporting the translation among object-oriented and semantically rich data models. It is extensible in the sense that new modeling constructs and constraints can be added by extending the primitives of ORECOM. This paper also describes the design and implementation of a data model and schema translation system which is being used to test the conversions between the constructs and constraints of data models such as IDEF-1X, NIAM, EXPRESS, and OSAM*. Applications of this system include schema sharing, database conversion, and semantics modification and extension of schema. The techniques and methodology developed in this system are also useful for schema integration and schema design.

Index Terms: Data Model Conversion, Schema Translation, Object-oriented Model, Semantic Model, Heterogeneous Database Management System, Interoperability

1. Introduction

* The initial support of this research was provided by the National Institute of Standards and Technology under grant# 60NANB7D0714 and the subsequent support has been provided by the Florida High Technology and Industry Council under grant# UPN91092323.

One of the main objectives of an integrated computing system is to allow data and the schemata of all component systems to be shared across multiple divisions of an enterprise or multiple enterprises. The data and schemata have been separately developed for supporting different aspects of applications using different computing and database systems. The main barrier to effective data and schema sharing is the diversity and heterogeneity of data models used by the various component systems. For example, in a manufacturing environment, data used to support various designs, process planning, and manufacturing activities are stored and processed by different file management systems, database management systems, and in-house software of all kinds. They have different logical as well as physical structures and are processed by different application programs which run on different hardware systems and operating systems. To share the data generated by different activities of the entire product life cycle starting from product design, analysis, process planning, manufacturing, sale, and service, a heterogeneous database management system needs to be developed to run on top of the existing data management facilities used by a manufacturing enterprise. The main task of such a system is to translate the logical (schemata) and physical (data) representations of the data residing on one component system into the representation suitable for use by another component system. The translation of logical schemata is essential for achieving data sharing among heterogeneous component systems. Problems and solutions related to many issues of interoperability among heterogeneous systems have been well addressed and documented in [HEIM85, LITW86, NSF89, SHET90, ALON91, GANG91, KENT91, URBA91, KRIS91, BARS92, BENN92].

Most of the existing work in logical schema translation has been limited to traditional data models (e.g., hierarchical, network, and relational model) [BILL79, VASS80, CARD80a,b, LARS83, HWAN83, etc.]. In today's complex and non-traditional applications such as CAD/CAM and integrated manufacturing systems, the transformation becomes

more difficult because the schemata available in different applications are defined by much more complex high-level object-oriented and semantic models such as E-R model [CHEN76], EXPRESS [SCHE89], NIAM [VERH82], IDEF-1X [LOOM86], OSAM* [SU89], and others surveyed in [HULL87, PECK88]. Each of these models has different modeling constructs (e.g., entity or class, object or instance, relation or association, etc.) for modeling the structural properties of an application. These constructs often have a number of semantic constraints associated with them. They are either explicitly specified by some keywords or rules (e.g., primary key, non-null, total participation, one-to-one mapping, etc.) or implicitly represented by the constructs (e.g., all instances of a subtype entity are also members of a supertype entity). Furthermore, some models (e.g., OSAM* and EXPRESS) also allow user-defined operations and rules. Different data models may name their constructs and constraints differently even though they may represent the same semantic properties. Others may have the same names but refer to totally different properties. Also, constructs in different models may share similar but not identical properties. It is important that all semantic properties should be preserved in a schema translation. Clearly, those translation algorithms introduced for the traditional data models are not suitable to handle the newer complex modeling concepts. Therefore, there is a strong need for a good technique to deal with the data model proliferation and heterogeneity problems and to achieve data sharing among heterogeneous systems.

In this paper, we present an approach which can perform the translation of complex data models in a general manner. The major contributions of this approach are: (1) the introduction of an object-oriented, rule-based, extensible core model (ORECOM) which is used to represent in a neutral form the semantic properties of diverse high-level modeling constructs and constraints, and (2) the effective schema translation technique and the management of semantic discrepancies in schema translations. This core model is a low-level neutral model which consists of general facilities for specifying the structural

properties, operations, and knowledge rules associated with application data. The modeling constructs, operations (or methods) and constraints available in a high-level data model can be decomposed and represented in ORECOM in terms of primitive constructs, operations and knowledge rules. Transformation between any pair of the existing models (ER, IDEF-1X, NIAM, OSAM*, EXPRESS, etc.) can be carried out at a lower level by matching their core model representations. If any discrepancy is found, the non-overlapping constructs, operations and rules can be used as a basis for generating an explanation of the discrepancy to the user and/or some program codes, which can be incorporated in the user's application program to account for the discrepancy.

This paper is organized as follows. In section 2, related research works are reviewed, and issues associated with the translation approaches are discussed. The proposed Object-oriented Rule-based Extensible COre Model (ORECOM) is presented in Section 3, in which we define its primitive structural constructs, operations, constraint types, and rules. In Section 4, the methodology and architecture of a data model and schema translation system based on ORECOM is given. Applications of this system and the significance of the R&D work are also described in this section. The status of an implemented prototype system is given in Section 5 together with an example of translating a NIAM schema to an equivalent IDEF-1X schema. Lastly, a conclusion is given in Section 6.

2. Review of Related Research Work

Most of the earlier work on data model and schema translation was developed in the early 80's and focused only on the translations among traditional data models (i.e., hierarchical, network, and relational models). Some of them can perform translations in either direction from any one of the three models to the others [BILL79, CARD80a,b, POTT84, GRAN84, JACO85, BREI86, CARD87]. Some can perform only specific translations such as the translations between the relational and network models [DEVO80,

KATZ80, LARS83], or the translations from the hierarchical or network model to the relational model [VASS80, HWAN83]. After semantic data models such as E-R [CHEN76] and its variations, NIAM [VERH82], Iris [FISH87], etc., were introduced, the data model and schema translation efforts were extended to cover these data models [VASS80, SHIP81, MORG83, BRIA85, FLYN85, SEN86, LYNG87, ELMA89, DEMU87,88, LEUN88, HSIA89, OZKA90, etc.].

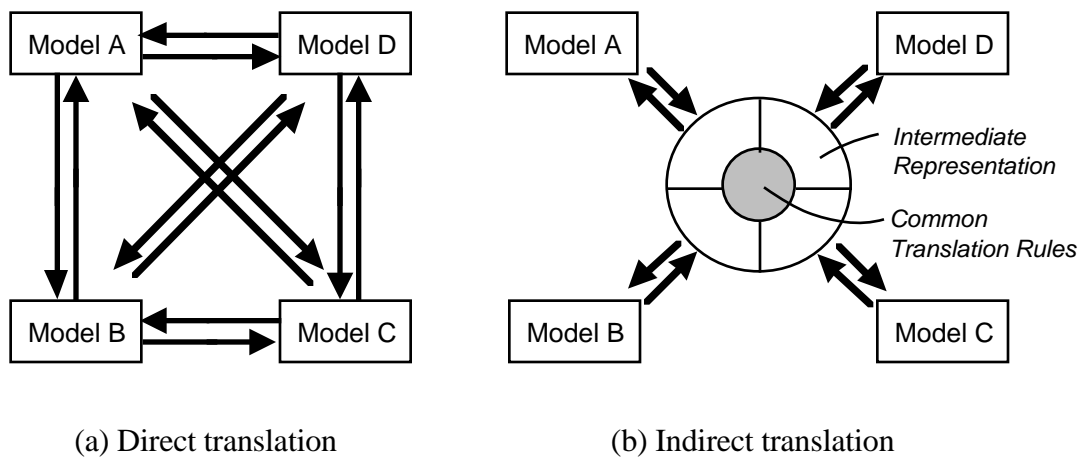


Figure 2.1 Two different translation approaches.

In general, the existing model/schema translation approaches can be categorized into two groups: direct translation or indirect translation. A direct translation, as shown in Figure 2.1(a), translates the schema defined by one model to that of other models directly by applying some dedicated translation rules or algorithms. The disadvantage of this approach is that the translation algorithms are pair-wise specific and can not be used for all pairs of source and target models. Therefore, the two mappings between E-R and the relational model described in [VASS80] need two different algorithms; one for each direction. Similarly, the mappings from E-R, EER, or E^2/R to the relational, hierarchical, or network

model [ELMA89,OZKA90], from Iris to the relational [LYNG87], from NIAM to the relational model or ACS [LEUN88, FLYN85], from the relational model to the network model or its reversal [LARS83], etc., all need specific translation algorithms. Since these direct translation algorithms are model specific, there is no uniform approach to support algorithm development for different data models. As a result, the existing algorithms are highly intuitive and rather complex and are entirely based on the system developers understanding of the source and target models.

Indirect translation approach, on the other hand, avoids the development of a large number of unsharable translation algorithms by using a common *intermediate model*. Instead of translating a source model to a target model directly, it divides the translation process into two steps as illustrated in Figure 2.1(b). The source model is first converted into the intermediate model which is in turn translated into the target model. The indirect approach has the following advantages: (1) **Low complexity** -- For N data models, the direct approach requires $N*(N-1)$ dedicated translation algorithms, while the indirect approach needs only $2*N$ algorithms, and (2) **High extensibility** -- For adding a new model into the translation system which handles N models, only two new algorithms need to be developed instead of $2*N$ algorithms.

An important decision to be made for using the indirect translation approach is the selection of an intermediate model. Different data models have been used in the existing work. The E-R model in HD-DBMS [CARD80a,b, 87], the functional model in Multibase [SMIT81], the ECR in DDTS [DEVO80], the IDEF-1X in IISS [IISS86], the attribute-based model in MLDS [DEMU87,88, HSIA89], the OSAM* in IMDAS [KRIS88], and the canonical model in PRECI [DEEN81] are just some examples. Other representations are also possible. For example, the first-order logic is used in [JACO85, GRAN84], a Logical Data Definition Language (LDDL) is developed in [BILL79], a hypergraph is

considered in [MORG83], and mathematically precise definitions are proposed in [KLUG78].

The selection of a proper intermediate model is important because it not only determines the capability of a translation system but also affects the correctness of the result of translation. In our opinion, there are three requirements for the intermediate model. Firstly, it has to be high in expressive power. If a source data model has some constructs or constraints that can not be captured by the intermediate model, the semantic properties it represents will be lost. The expressive power of the intermediate model has to be at least as strong as the sum of all the data models to be handled by the translation system. However, if we design an intermediate model to incorporate all the structural constructs and constraints of all these data models, the resulting model will be too complicated to use. Besides, it is difficult for a single model to effectively incorporate in a "seamless" manner all the existing modeling constructs having different complexities and incompatible features. For this reason, the intermediate model should provide a low-level primitive semantic representation which serves as a neutral representation into which the high-level constructs of diverse data models can be translated. The approaches taken in [GRAN84, JACO85, SEN86, DEMU87, DEMU88, HSIA89] are the right direction in meeting this first requirement. However, their models can not be easily used as the intermediate model for the neutral representation of object-oriented data models. In our work, we use an object-oriented model as the intermediate model. It provides very primitive and general structural constructs and very powerful knowledge rule specification facility to capture diverse semantic properties of the existing data models.

Secondly, the intermediate representation of semantic properties has to be explicitly enough so that any fine discrepancy between a source modeling construct/constraint and a target modeling construct/constraint can be explicitly identified and be used to generate either an explanation to the user concerning the discrepancy or program code to handle the

discrepancy. Otherwise, semantic preserving translations can not be guaranteed, and an incomplete or incorrect target model or schema may be generated by the translation system. In our work, "micro-rules" are used to define the various semantic properties explicitly. Lastly, the intermediate model has to be extensible in the sense that its modeling power can be extended to capture the new semantic properties introduced in some additional data models to be included in the translation system. Without this capability, the translation system will not have a general utility in a heterogeneous computing environment. Unfortunately, none of the existing translation systems and their intermediate models provide the extensibility feature. In our work, an extensible model is used as the intermediate model. Extensibility is achieved by user-defined data types, associations, operations, and knowledge rules which capture structural, procedural and declarative semantics found in many existing data models.

3. An Object-oriented Rule-based Extensible Core Model (ORECOM)

ORECOM consists of a small number of primitive structural constructs (objects, classes, and associations), an operation specification component, and, most importantly, a very powerful rule specification component. It allows the semantic properties of data models to be explicitly defined in terms of these primitive constructs, operations and rules. Thus, it provides a general and uniform framework for modeling objects, associations of objects, operations, and semantic constraints found in an application world.

3.1 Primitive Structural Constructs

There are three kinds of structural constructs defined in ORECOM: object, class, and association. We will use an example schema shown in Figure 3.1 to illustrate these basic constructs.

Object. All the real world abstract or physical entities, events, processes, and functions etc., are uniformly treated as objects in ORECOM. An ORECOM object may be self-named

(e.g., an integer "15" as a circuit number, a string "JX66a" as a pump type, etc.) or system-named (e.g., a pump, a hydraulic circuit, etc.). A self-named object uses the value that it represents as the identifier, while a system-named object is referenced externally by a user-defined key and internally by a system-generated unique identifier. For example, we can distinguish two integer objects by their integer values, but, for two pumps, we use a key attribute such as pump-id and two different values for the user of a database system to distinguish the pumps and two system-assigned identifiers for the system to distinguish them internally. Another important characteristic of an object is its structure. In ORECOM, objects are allowed to be aggregated together using data structure like set, bag, list, array, or tuple, etc., to form a new object. Therefore, an object can be simple or composite. Effectively, a composite object can be decomposed into simpler structured objects. For instance, the object "list (1, 2, set (3, 4))" is formed by three ordered objects: "1", "2", and "set(3,4)".

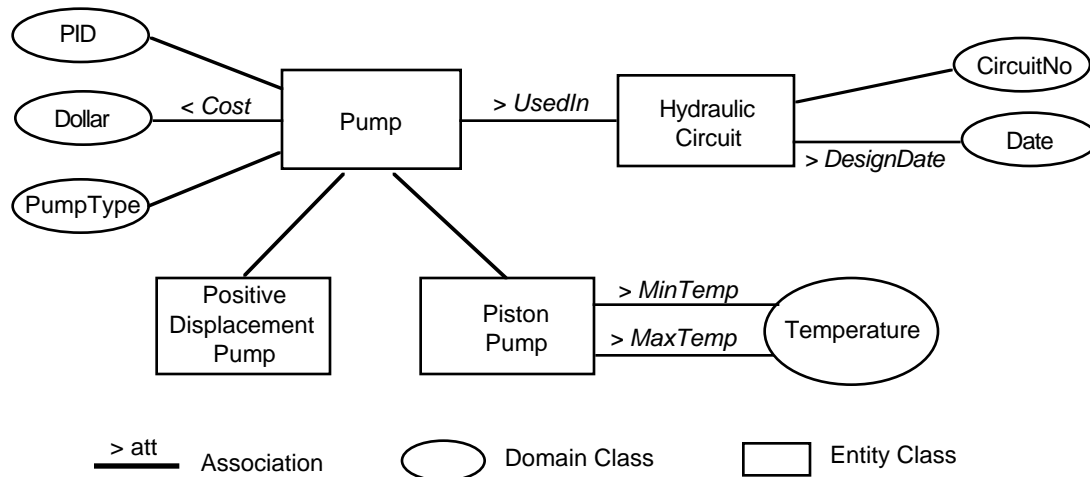


Figure 3.1 Example of an ORECOM schema.

Class. A class is a semantic declaration and a grouping of a set of homogeneous objects. The declaration of a class (i.e., the intension of a class) includes a class name, its associations with other classes, the operations and micro-rules that can be applied to its

objects. Objects that belong to the same class (i.e., the extension of a class) share the same declared characteristics. ORECOM distinguishes two types of classes: domain classes and entity classes. Domain classes contain self-named objects and entity classes contain system-named objects. In Figure 3.1, the rectangular boxes represent entity classes and the ovals represent domain classes.

Association. An association is a binary relationship between two classes. Two classes are associated with each other if they are structurally related and have some semantic constraint(s). For example, the Pump and the HydraulicCircuit class in Figure 3.1 are associated with each other. This association is named by ">UsedIn", where the sign ">" indicates that this name is an attribute of Pump and HydraulicCircuit is the domain of this attribute. An inverse name of the same association is implied (i.e., an association is always bi-directional). We can add "<Contains" on the same association as the inverse of ">UsedIn". Named associations are useful especially when there are multiple associations between the same pair of classes (e.g., the minimal temperature and maximal temperature relationships between PistonPump and Temperature). However, the specification of association name is optional if there is no ambiguity on the identity of an association such as the association between classes Pump and PumpType.

In ORECOM, the semantic constraints (e.g., cardinality, inheritance, etc.) of an association are captured by micro-rules and macros (to be described below). For example, Pump in Figure 3.1 is intended to be a superclass of PistonPump although this 'superclass/subclass' relationship is not explicitly shown in the schema. This 'superclass/subclass' relationship is represented in ORECOM by (1) an association between Pump and PistonPump, and (2) a set of macros which represent the semantic properties and

constraints defining the superclass/subclass relationship such as inheritance¹, cardinality, participation, etc.

N-ary associations, which are recognized by some high-level data models such as ER and OSAM*, are represented structurally in ORECOM by a number of binary associations. The additional semantic properties or constraints of n-ary associations, which are not captured by the binary representation, are captured by micro-rules and macros in the same manner as we have described above for a binary association.

Using the above primitive structural constructs, the structural properties of many existing data models can be represented in a neutral form. For example, a tuple in the relational model would be represented by the associations between the object that corresponds to the entity that the tuple represents and the domain objects that represent the attribute values. The entities and relationships of the E-R model are represented as objects and their associations. Objects in a supertype and a subtype in EXPRESS are represented as associations between the objects of these classes having the same object identifiers but different instance representations.

3.2 Primitive Behavior Constructs

The behavioral properties of objects and object associations are captured in ORECOM by system-defined and user-defined operations and micro-rules. The system-defined operations of ORECOM are operations for accessing or manipulating objects and their associations. ORECOM also allows user-defined operations to be defined (in class declaration) to capture the procedural semantics represented by methods of an object-oriented source or target model. Additionally, ORECOM provides a rule specification

¹Inheritance is not treated as a structural construct of the core model because many existing data models do not support inheritance. In order to represent the inheritance in these models, it has to be treated as other constraints and represented by using rules which will be described in the next section.

facility to allow the semantic constraints associated with objects and associations to be declaratively specified.

In a translation between an object-oriented model and a non-object-oriented model, a well-known problem is the mapping of user-defined operations which are encapsulated in the object-oriented model. In the traditional O-O paradigm, a user-defined operation consists of two parts: (1) a method specification or signature which specifies the name of a method, the parameters it has, and the data type of a returned value if one exists, and (2) a method implementation or program code. Since any arbitrary computation (i.e., procedural semantics) can be embedded in the method implementation and there is no known technique for analyzing a piece of program code to uncover the intended semantics of the programmer, the best a model and schema translation system can do is to convert the method specification (the signature) of a source model to its intermediate representation in the neutral model and either leave the method implementation (the program code) alone or translate it into a program in some programming language desired by the user of a target model and system. (The latter approach is a programming language translation problem which is not what we are concerned with in this work.) Therefore, in our work, if an object-oriented schema is to be translated into a target schema which is not object-oriented, the user-defined operations or methods captured in the neutral model will be shown in the output as a discrepancy between these two schemata. The user of the target schema can take the implementation and specification of these user-defined operations or methods as a reference or guide in his/her application development. If some of the semantic properties that are hidden in a method implementation can be specified in terms of constraint rules (i.e., the method implements some constraints), these constraint rules together with the source schema can be used as the inputs to the model/schema translation system to produce the target schema as to be described in Section 4.2.

Object Operations. ORECOM supports five system-defined object operations: Create, Destroy, Associate, Disassociate and Dot operation. These five object operations are low-level, primitive operations that can be performed on objects and their associations. Comparing with the general database operations like Insert, Delete, Update, and Retrieve, the ORECOM object operations are more primitive operations. For example, a database operation such as an Insert is decomposed into a Create operation to generate an object and a number of Associate operations to associate a number of objects, which serve as the attribute values, with the generated object. An Update operation is represented in ORECOM by a Disassociate operation of an object with an object that represents an old value and by an Associate operation of the object with a new value. Thus, Insert and Update operations share a common primitive Associate operation.

The reason for using these primitive operations is as follows. Since different constraints can be associated with different attributes and domains associated with the definition of a class, and since we will use the object operations as part of the trigger conditions to define micro-rules (to be described next) to specify the constraints, the object operations must be primitive operations so that every micro-rule deals only with the checking of a very fine and specific constraint. By decomposing high-level modeling constructs and constraints into their micro-rule representations, micro-rules can be used as the fundamental semantic units to identify the similarities and differences among these high-level modeling constructs and constraints. Thus, the commonalities and subtle differences underlying different database operations and their constraints can be detected and accounted for.

In the following explanation of the primitive object operations, we shall use lower case characters (e.g., x, y, z) to represent objects and upper case characters (e.g., X, Y, Z) to represent classes.

- Create The operation 'x.Create' generates a new object x in class X².
- Destroy The operation 'x.Destroy' deletes an existing object x from the class X.
- Associate 'x.Associate(α , y)' is an operation which links object x and object y through an association named ' α ' between class X and Y, where ' α ' is an attribute of X and Y is the domain of ' α '.
- Disassociate 'x.Disassociate(α , y)' is an opposite operation of Associate, which breaks the link α between x and y, if that link exists.
- Dot operation The operation 'x. α ' returns all objects in class Y (which serves as the domain of attribute α) that are directly associated with object x. We can also specify 'x. α . β ' which returns all Z objects (where class Z is the domain of attribute β of class Y) that are indirectly associated with x through some Y objects.

Micro-Rule. The behavioral properties of high-level data models are captured in ORECOM by a number of primitive object operations and a set of what we called micro-rules that govern the operations. A micro rule is defined in two parts: a trigger condition and a rule body. The trigger condition specifies a trigger time which can be "before", "after" or "in parallel", and an object operation described above. The rule body is specified as a well-formed-formula (*wff*) of an object calculus[KAME92], which is evaluated to be either true or false when the rule is triggered. The object calculus is a pattern-based first-order logic used as the formalism for specifying the semantics of object-oriented databases. As an example, a micro-rule for class X can be defined as follows:

$$\text{After } x.\text{Disassociate}(\alpha, y1) : \exists y2 (x * > \alpha y2)$$

²The assignment of an object identifier to this created object is presumably done by an object-oriented system.

where x , y_1 , and y_2 are called object variables which range over class X and Y , respectively. This rule has a trigger condition "After x .Disassociate(α , y_1)" and a rule body " $\exists y_2 (x \text{ *}>\alpha y_2)$ ", which is a quantified *wff*. Together it says that after an object x is disassociated from object y_1 (i.e., breaking the association α between these two objects or delete the α value of x), object x must still be associated ($\text{*}>\alpha$) with some other Y object(s) through the same association. Otherwise, the rule fails and the operation x .Disassociate(α , y_1) is rejected. The "*" sign is the association operator used in the calculus to specify the association of objects. This micro-rule can be used to define the semantics of "non-null" associated with α value of X (i.e., every X object has to have an α value or every object in class X must always associate with some Y object(s) through the association α). A constraint specification language based on the similar formalism has been reported in [SU91].

After having analyzed a number of semantically rich data models including IDEF-1X, NIAM, EXPRESS, E-R, OSAM*, etc., we have come to the conclusion that various types of semantic constraints captured by these high-level models can be very explicitly represented by micro-rules of the type described and illustrated above. This is so because all database constraints are concerned with what a database management system should check and do if some operation is to be performed on a database either for semantic integrity or security reasons.

The basic idea of our rule-based model/schema translation is therefore to decompose the modeling constructs of high-level data models into a general and primitive structural representation and those semantic properties not captured by the structural representation are represented by micro-rules with triggers. Comparisons between high-level modeling constructs can be carried out by matching their neutral representations. Their discrepancies can be explicitly represented by the primitive structures and micro-rules that failed to match in the comparisons.

3.3 Macros

A high-level construct or constraint is generally represented by a set of micro-rules with different triggers. If all the high-level constructs and constraints of the data models to be handled by a model/schema translation system are directly mapped to micro-rule representations, comparisons of these representations can be very difficult to carry out either manually or automatically. Since many types of semantic constraints are commonly available in the existing data models, an intermediate level of representation in the form of what we call "macros" would ease the comparisons.

A macro is a constraint type that can be applied on a given set of classes. It can have a number of options which are specified as parameters of the macro and represent some finer semantic properties of the constraint type. Each macro with some selected options corresponds to a set of micro-rules. Semantic representations at the macro-level are more compact than at the micro-rule-level and are easier for the designer and developer of the translation system to understand the distinctions of modeling constructs.

In our work, we have analyzed a number of existing semantically rich data models including IDEF-1X, EXPRESS, NIAM, and OSAM*. The modeling constructs of these models are manually transformed into ORECOM's structural representations in terms of object classes and their associations, and the modeling constraints recognized by these models are analyzed to find their common and primitive properties. The analysis was guided by first considering all constraints associated with the membership of an object class, then those associated with the binary association between two object classes, and finally those associated with multiple binary associations (or an n-ary association) among classes. Seven macros or constraint types have been identified and shown in Figure 3.2. For space reason, we shall briefly describe six of these macros and use the macro called Participation as a detailed example to illustrate the concept of macro and micro-rule.

All data models have the concept of membership, i.e., the grouping of objects (physical objects, abstract things, events, functions, processes, etc.) having similar properties. In ORECOM, we call this grouping a class, which is specified by an object type (self-named or system-named objects), an object structure (e.g., set, list, bag, array, etc.), some user-defined identifier(s) (i.e., key attribute(s)), a class type (defined by methods) and some membership constraint(s) (e.g., scalar or range values). The cardinality constraints refer to the 1-1 (one to one), 1-N (one to many), N-M (many to many), and k_1 - k_2 (where k_1 and k_2 are positive integer numbers) mappings between objects of two directly or indirectly associated classes. The inheritance property in a superclass-subclass or IS-A association between two classes, which is recognized by some data models, is represented as a constraint type (i.e., macro) in ORECOM. The privacy constraint, which is recognized by a limited object-oriented models, restricts the access of object properties and operations to some associated objects (e.g., the private, protected and public accesses in C++). The constraint types 6 and 7 in Figure 3.2 specify the value and logical relationships among objects of an n-ary association using some mathematical formula and logical expression, respectively. An example of type 6 is that the attribute value of Y3 is derived based on the values of Y1 and Y2 as shown in the figure. The example for type 7 states that if an object of class X is not associated with any object of Y1, then it must be associated with either an object of Y2 or Y3. Each of these constraint types has a number of options and each option corresponds to a set of micro-rules that semantically define the specific constraint.

1. Membership

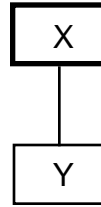
- What kind of objects can be members of class X?



- object type** (self-naming?)
- object structure** (composite?)
- identifier** (has key?)
- class type** (parent?)
- membership constraint** (restricted?)

2. Participation

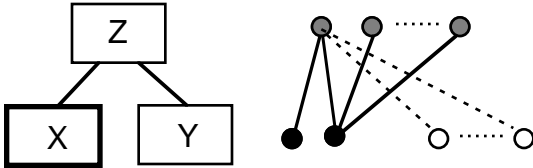
- How many X objects should participate in this X-Y association?



- total participation**
- partial participation**
- or
- limited by a particular range**

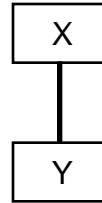
3. Cardinality

- How many Z (X) objects can be associated with each X (Z) object?
 - How many Y objects can be indirectly associated with each X object through some Z?



4. Inheritance

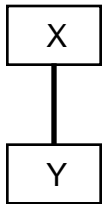
- Can Y objects inherit the associations and operations from their associated X objects?



If yes, X is a super-class of Y, and Y is a sub-class of X.

5. Privacy

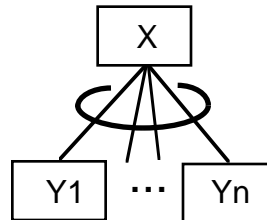
- Is X-Y a private association?



If yes, the operations and associations (including X-Y itself) of X (Y) class are accessible to Y (X) objects only.

6. Mathematical Dependency

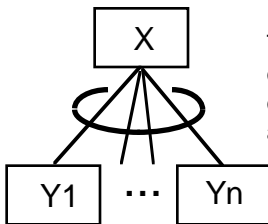
" $Y_3 = (Y_1)^2 + (Y_2)^2$ " or " $(Y_1 - Y_2) > Y_3$ "



The associations of an object of X with objects of Y1, ..., Yn must satisfy a given mathematical expression or formula.

7. Logical Dependency

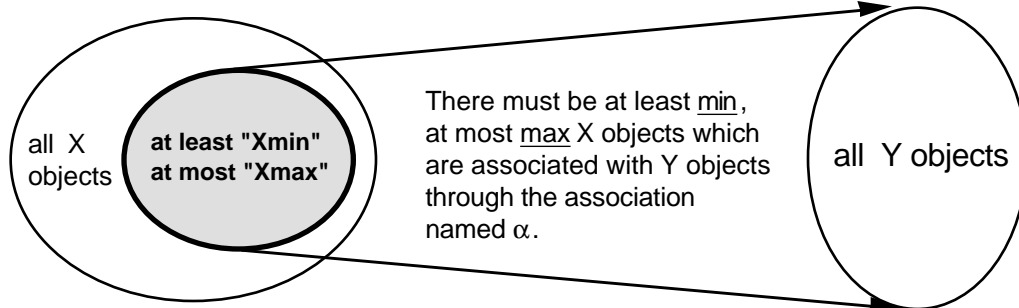
"IF (X ! Y1) THEN (X * Y2) OR (X * Y3)"



The associations of an object of X with objects of Y1, ..., Yn must satisfy a given logical expression.

Figure 3.2 ORECOM macros.

Participation (X, min, max, α , Y)



(a)

Trigger Condition		Rule Body
after	x.Create	IF min = 'all' THEN $\exists y (x * > \alpha y)$
before	x.Associate(α , y)	Count({ x $\exists y (x * > \alpha y)$ }) < max
before	x.Disassociate(α , y)	Count({ x $\exists y (x * > \alpha y)$ }) > min

(b)

Figure 3.3 (a) Participation macro.

(b) Micro-rule representation of Participation macro shown in (a).

The semantics of the second macro called Participation in Figure 3.2 is illustrated in Figure 3.3(a). To simplify the explanation of this macro, we consider only the participation constraint between two classes, which is a special case of the general format of Participation macro involving an n-ary association between a class and a set of other classes. The Participation macro in Figure 3.3(a) allows both lower and upper limits to be specified for a class, X, to restrict the total number of X objects that can participate in the X-Y association with Y objects. It has five parameters³: X, min, max, α and Y, as shown in Figure 3.3(a). The X and Y are class names. α is the name of the association between X and Y. It also specifies

³In the general form, we allow the constraint defined in each macro to be applied to not all objects of each participating class in the macro, but only those objects which satisfy some predicates. Therefore, the formal definition of the Participation macro contains predicate parameters for every participating class to select some qualified objects which are further constrained by the Participation macro.

that α is an attribute of X and its domain is Y. The min and max specify that there must be at least "min" and at most "max" X objects associated with Y objects through attribute α .

The behavioral semantics of this Participation macro can be represented by the micro-rules shown in Figure 3.3(b). The first micro-rule is used in the case of a total participation (i.e., min equals "all"). In this case, whenever an object x is created, it must associate with some Y object(s) through attribute α . Other options of this constraint type are to allow the specification of the minimal and maximal number of X objects involved in the association. They can be expressed by the second and third rules in Figure 3.3(b). The second rule is fired whenever an X object is associated with a Y object through α to make sure that the upper limit of participation of X class is not exceeded. Similarly, the lower limit is enforced by the last micro-rule.

To demonstrate the use of Participation macro, let's assume in Figure 3.1 the HydraulicCircuit class has a non-null attribute CircuitNo defined on the domain class CircuitNo. (Semantically, this is equivalent to say that the objects of HydraulicCircuit are totally participated in the association named CircuitNo with the objects of CircuitNo class.) It is noted that the attribute name is assumed to be the same as its domain. This constraint can be represented as:

Participation(HydraulicCircuit, **all**, all, CircuitNo, CircuitNo).

On the other hand, if a constraint specifies that there must be hydraulic circuits available of at least ten different circuit numbers, then the minimum participation of CircuitNo objects would be ten. Using the Participation macro again, this constraint can be represented as

Participation(CircuitNo, **10**, all, belongs_to, HydraulicCircuit)

where 'belongs_to' is the inverse attribute of 'CircuitNo'.

We note here that the semantics of "non-null" is the same as one aspect of the Participation constraint and they are grouped in the same macro. We note also that the semantics of "primary key" in the relational database has the "non-null" property which is in turn represented by one of the primitive concepts of Participation. Similarly, many other semantic constraints found in high-level data models can be decomposed and represented by various combinations of the macros shown in Figure 3.2 and their options.

3.4 The Extensibility of ORECOM

If a neutral model of a model and schema translation system captures only a fixed set of semantic properties, it will not be able to accommodate any special semantic property that an existing or new data model may have. As a result, the translation system will only work for a fixed number of pre-determined data models. On the other hand, if the neutral model is extensible, new semantic properties can be added by increasing the modeling power of the neutral model itself to capture the new semantics. Naturally, the extension of the neutral model should not entail changes to the translation algorithms and the overall framework and strategy of the translation system.

The extensibility of ORECOM is accomplished by user-defined data types, user-defined operations, user-defined association types, and user-defined micro-rules. As an object-oriented model, it supports user-defined data types and operations as the traditional object-oriented paradigm. In addition, ORECOM supports user-defined association types through the use of macros and micro-rules. Instances of association types are defined as first-class objects in ORECOM. Based on its semantics, a user-defined association type can be defined as a class having a new combination of existing macros with proper options. In case that the current set of macros are not sufficient for capturing the semantics of a user-defined association type, new macros can be defined by using either the existing micro-rules and/or user-defined micro-rules (expressed in the object calculus). Micro-rules with triggers

can be used to specify any added semantics. A schema translation system that performs its translations based on these macros and micro-rules is therefore able to accommodate the added semantics.

A significant advantage of using an extensible neutral model is that the translation system (to be described in the next section) is also extensible. Therefore, the translation system is a general system in the sense that it can be extended to accommodate upgraded existing data models as well as new data models in the future without affecting the translation algorithm and the general translation strategy.

4. The Design and Applications of a Data Model and Schema Translation System

Since a schema is defined in terms of the modeling constructs of an underlying data model, the translation between schemata will need the equivalence mappings between the constructs of the two data models. The ORECOM-based translation system therefore consists of two subsystems. A data model translation subsystem derives the equivalence mappings among modeling constructs and the schema translation subsystem uses the mapping information to perform the translation. In this section, the architecture of these two subsystems and the process they performed are presented. Applications of this translation system including schema sharing, data conversion, and semantics modification and extension will be addressed. In addition, the impact of techniques and methodology of the system on the areas such as schema integration and schema design will also be described at the end of this section.

4.1 Subsystem-1: Data Model Translation

Data model translation is performed at the intermediate level of ORECOM representation. In this neutral and low-level representation, the similarities and differences between the source and the target models can be clearly identified. Procedurally, a data model translation consists of three steps which is illustrated in Figure 4.1:

- Step 1: Identify the basic modeling constructs and the associated constraints of both the source and target data models. (**Data model analysis**)
- Step 2: Determine the macro and micro-rule representation of each construct and the associated constraint(s) identified in Step 1. (**Decomposition**)
- Step 3: Based on these representations, determine the equivalent target representation for each source construct and its constraint(s) and the discrepancies if they exist. (**Equivalence analysis**)

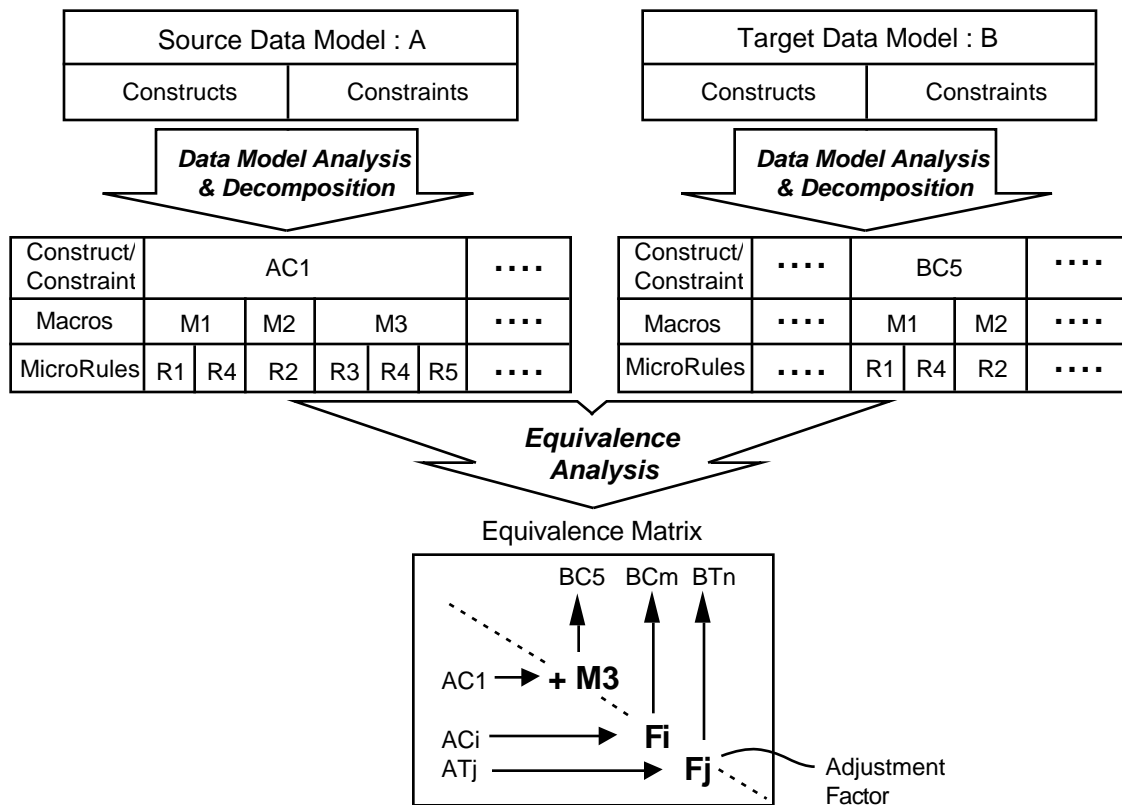


Figure 4.1 Data model translation.

As shown in Figure 4.1, the constructs and constraints of a source data model A are to be translated into those of a target data model B. The first two steps are executed by the 'Semantic Analysis' component of subsystem-1. (See Figure 4.3.) The result of semantic analysis is the decomposed constructs and constraints in the forms of macros and their corresponding micro-rules which are stored as the metadata of each processed data model. The final output of data model translation is an equivalence matrix generated by the component 'Equivalence Analysis' of subsystem-1. The rows and columns of this matrix represent the source and target modeling constructs and constraints. Each element of the matrix is an adjustment factor in terms of macros or micro-rules, which indicates the discrepancies between a source construct and a target construct. By looking at the elements of a row, an equivalent (i.e., no adjustment factor) or the closest (i.e., the least number of adjustment factors) target construct can be found for a given source construct. As an example, let us consider the construct AC1 of model A and BC5 of model B. In Figure 4.1, AC1 is decomposed after step 2 into three macros, M1, M2 and M3. BC5 is decomposed into M1 and M2. If BC5 is the closest construct of model B to AC1, then the equivalence information for AC1 is "AC1 = BC5 + M3", where "+ M3" is the adjustment factor for BC5 to be equivalent to AC1. This matrix entry states that AC1 is semantically equivalent to BC5 only if the macro M3 is added to BC5. We note here that, although we are using an equivalence matrix for the mapping of each pair of data models, this approach is different from the direct translation approach discussed in section 2 since the translation algorithm is separated from the data (i.e., the mapping information provided by the matrices) that drive the translation process. The translation algorithm is general and equivalence matrices are specific to pairs of data models.

4.2 Subsystem-2: Schema Translation

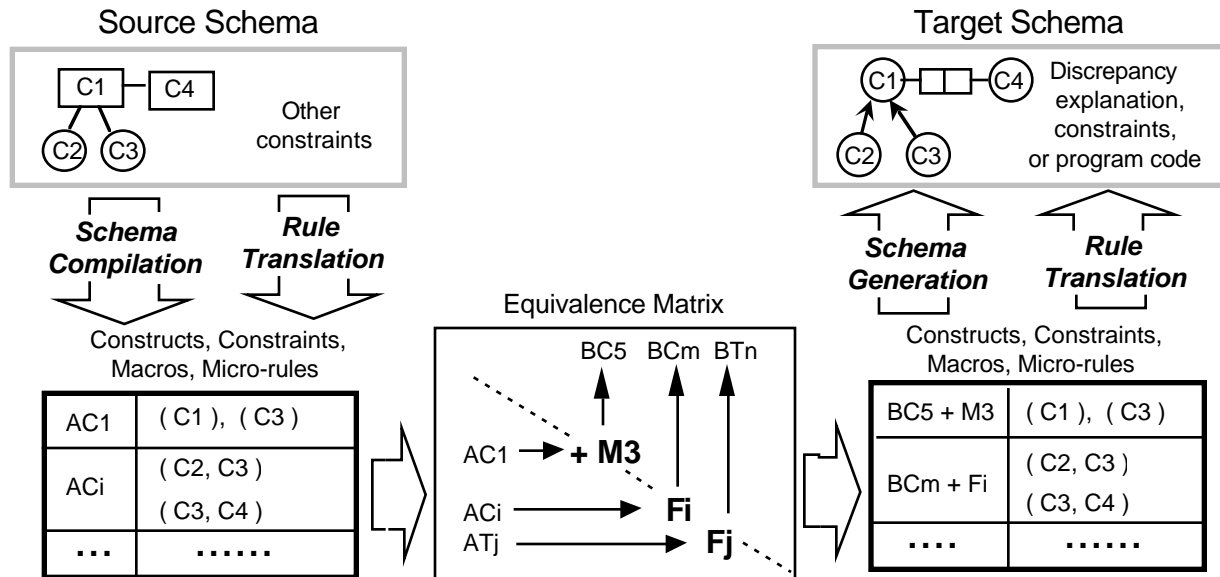


Figure 4.2 Schema translation.

Schema translation is driven by the equivalence information generated by subsystem-1. Given a source schema, its equivalent target schema defined in a different data model is generated by the following three steps:

- Step 1: Compile the input schema into the representation of the construct and constraint types of the source data model which have been identified in Step 1 of subsystem-1. (**Schema compilation**)
- Step 2: For each construct and its associated constraint(s) of Step 1, search the equivalence matrix for its equivalent construct and associated constraint(s) in the target data model. (**Equivalence search**)
- Step 3: Collect all the identified constructs and constraints of the target data model and generate the target schema. If there are discrepancies found in Step 2, they are used to either generate an explanation to the user so that he/she can take care of the discrepancies in his/her application development using the target schema or generate program codes to be incorporated into the user's application system. The automatic generation of explanations and programs is possible since the semantics

of the discrepancies have been explicitly defined by macros and micro-rules which define the trigger conditions and database operation in great details. (**Schema generation**)

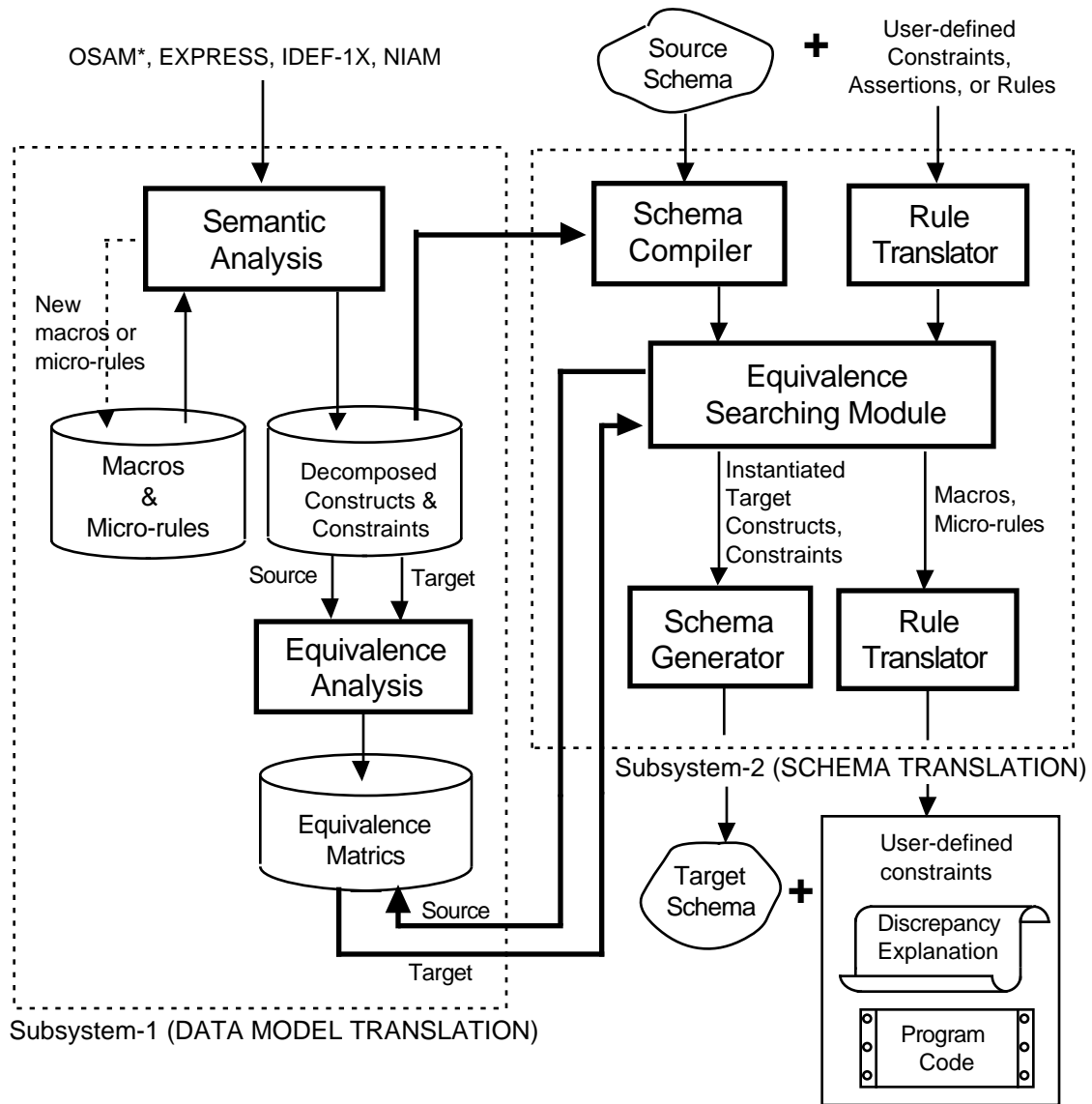


Figure 4.3 Architecture of data model and schema translation system.

The overall procedure of schema translation is illustrated by Figure 4.2. The architecture for implementing this subsystem is shown in Figure 4.3. In this figure the relationship between these two subsystems is also shown. In subsystem-2, one of the inputs is the schema defined in a source data model. Another (optional) input is the augmented constraints or assertions, which represent the additional semantics that can be specified by the user to modify the source schema. The augmented constraints are a part of the semantic properties of the applications that were not modeled in the source schema⁴ but the user wants it to be incorporated in the target schema. These augmented constraints are translated by the Rule Translator into macros and/or micro-rules and merged with the compiled source schema.

4.3 Applications and Significance of the Translation System

Since the need of data sharing across multiple enterprises using different database systems is growing, the above research and development effort on data model and schema translation system is important and timely for achieving interoperability among heterogeneous systems. Any organization which is developing applications in an integrated heterogeneous computing environment will definitely benefit from the translation system. Under this environment, some applications of the translation system are as follows:

- **As a tool for schema sharing.** In an application domain such as Computer Integrated Manufacturing (CIM), the schema translation system can be used directly for schema (e.g., PDES product model) sharing.

⁴Some augmented constraints may be extracted from the code embedded in application programs or method implementations of an object-oriented model or in the rule specification block of a schema defined in a data model like EXPRESS. Or it can be obtained directly from the original schema designer or the schema translation administrator.

- **As a tool to support data conversion.** The first step in the conversion of data among systems in an integrated heterogeneous database system is the translation of the database schemata of the participating systems (in general they can be specified in different data models) into equivalent schemata supported by the integrated DBMS. Two cases are possible:
 - (a) The participating systems have well-defined schema and underlying data model (e.g., a vendor-supplied database such as an Oracle database). In this case, the well-defined database can be made accessible to an integrated DBMS such as IMDAS [KRIS88] by first applying the ORECOM translation system to translate the schema into an equivalent one supported by the integrated DBMS. Then, the conversion of data can be carried out using the integrated DBMS.
 - (b) Some software product does not have a well-defined schema and an underlying data model. In this case, ORECOM can be used to model the input and output structures of the software product so that it will have a well-defined interface. The ORECOM-based translation system can then be used to automate or assist in the translation of schemata and data between the software product and the rest of the integrated database management system.
- **As a tool for semantics modification and extension.** Used as either a schema sharing tool or as a support for data conversion, an enterprise may want to modify or extend the semantics captured by some existing schema defined by, say, a relational model before converting it to EXPRESS. The changed or added semantics can be defined by ORECOM's macros and/or micro-rules that can be incorporated into the relational-model-to-EXPRESS translation process.

In addition to the above schema translation applications, the technique and methodology that are used in developing the schema translation system have also significant impact on other related areas such as schema integration and schema design.

- **Schema integration.** Suppose an enterprise has some schema modeled by IDEF-1X and others modeled by the relational model. An individual is interested in an equivalent, integrated schema specified in EXPRESS. The translation system can be used first to convert the schemata in IDEF-1X and the relational model into the uniform ORECOM representation and then be integrated at that level (by applying some integration method) before being used to generate the corresponding EXPRESS schema. The conflicts, if exist, among schemata can be resolved or identified in ORECOM's macros and/or micro-rules as part of the integrated results so that no semantics is lost during the integration.
- **Schema design.** The techniques used in the schema translation system can also be used as a basis for a schema design tool which possesses the following capabilities: *automatic schema generation*, *schema verification*, and *schema optimization*. In such a schema design tool, a schema designer needs to specify only the application semantics and constraints in the form of macros and micro-rules. An enhanced schema generator should be able to automatically generate a schema in a desired data model based on the specification. Alternatively, a schema compiler can be used to translate *any* given source schema into ORECOM's low-level macro and micro-rule representation which precisely capture all of the semantic properties and constraints specified in the given schema. Since the decomposed representation of the schema is uniform, general verification and optimization algorithms can be developed to detect conflicting constraints and to remove redundant constructs and constraints.

5. System Implementation and An Example

In this work, we focus our analysis of data models on IDEF-1X, NIAM and EXPRESS, which have been studied by the standard community of PDES/STEP, as well as our own OSAM* model. We have manually converted the modeling constructs and constraints into their ORECOM representations and derived the pair-wise equivalence

matrices for all the combinations of these models. Although, subsystem-1 can be automated, we have not done so at this stage. Program development for this part is in progress. It is important to know that the analysis of models and the generation of equivalence matrices are one-time tasks. Unless, some model is changed, the matrices need not be modified or regenerated. We make use of the manually derived equivalence mappings to implement subsystem-2. Subsystem-2 has been implemented in C running on both UNIX-based Sun workstations and AIX-based IBM RS-6000 workstations. The translation system has a graphical user interface (implemented in X-Window using OSF/Motif widget set) which is used for graphical input and output of IDEF-1X, NIAM and OSAM* schemata. The graphical representation of EXPRESS schema (EXPRESS-G) is being worked on by several organizations but is not readily available at this time. The implemented system is driven by the equivalent matrices which are provided to it as data. The prototype system has been demonstrated at RPI, NIST and IBM Kingston in May 1992. We are in the process of analyzing the relational model and the E-R model and their constraints for incorporation into the system. To incorporate these two models, additional equivalence matrices need to be formulated. However, Subsystem-2 itself needs not be modified.

In the remainder of this section, we shall give a simple translation example using NIAM to IDEF-1X conversion to illustrate the translation process based on the methodology and system architecture described in the last section. We shall assume that the reader is familiar with some basic constructs of these two models. The translation will be done from a simple NIAM construct to an equivalent IDEF-1X construct through the ORECOM representation. The source construct defined in the NIAM model is shown in Figure 5.1(a), which contains two object types, Pump (a non-lexical object type or NOLOT) and PumpType (a lexical object type or LOT). They are connected by a pair of role names in the two boxes. (To simplify the illustration, the role names are not shown in the schema.) The cardinality between Pump and PumpType is one-to-many as represented by the line

segment on top of the role name of Pump. A 'role-total' constraint denoted as "V" is added to the PumpType, which requires that every PumpType object has to be connected to some Pump object(s). It is not necessary, on the other hand, for the Pump objects to be connected to some PumpType objects. In other words, there are some pumps whose type is not known, but for each existing type we need at least one pump belonging to that type. The resulting target schema in IDEF-1X is shown in Figure 5.1(b), which will be explained later.

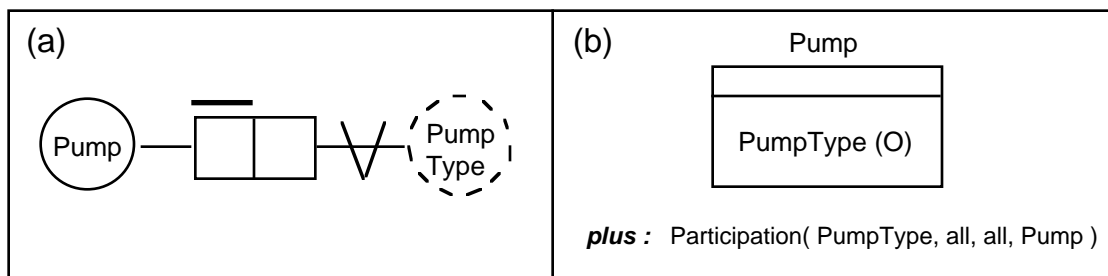


Figure 5.1(a) A NIAM source schema to be transformed.
 (b) The generated target schema in IDEF-1X model.

5.1 Data Model Translation: NIAM to IDEF-1X

As described before, data model translation involves three steps: data model analysis, decomposition, and equivalence analysis. Part of the result of data model analysis for NIAM and IDEF_1X is listed in Figure 5.2. The constructs and constraints of NIAM, designated as Nc.1, Nc.2, Nc.3, and Nt.3.1, represent the LOT, NOLOT, bridge with one-to-many constraint, and role-total constraint, respectively. The closest corresponding constructs in IDEF-1X are attribute (Ic.1), entity (Ic.2), and relation between an attribute and an entity (Ic.3).




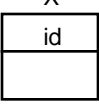
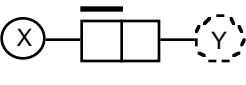
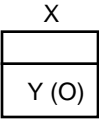
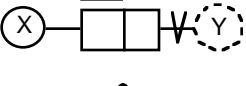
NIAM		IDEF-1X	
Construct/Constraint	Macro Representation	Construct/Constraint	Macro Representation
Nc.1 	Membership (X, SN, no id, S, T, C)	Ic.1 	Membership (X, SN, no id, S, T, C)
Nc.2 	Membership (X, NSN, id, S, T, C)	Ic.2 	Membership (X, NSN, id, S, T, C)
Nc.3 	Participation(X, 0, all, Y) Participation(Y, 0, all, X) Cardinality(X, 1, 1, Y) Cardinality(Y, 1, N, X)	Ic.3 	Participation(X, 0, all, Y) Participation(Y, 0, all, X) Cardinality(X, 1, 1, Y) Cardinality(Y, 1, N, X)
Nt.3.1 	Participation(X, 0, all, Y) Participation(Y, all, all, X) Cardinality(X, 1, 1, Y) Cardinality(Y, 1, N, X)	⋮	⋮

Figure 5.2 (a) Macro representation of NIAM constructs and constraints.
(b) Macro representation of IDEF-1X constructs and constraints.

The next step (i.e., decomposition) is to determine the ORECOM's macro representation for each identified construct and its associated constraint(s). For example, the X in Nc.1 is a lexical object type (LOT), when mapped into ORECOM, it becomes a class containing self-naming (SN) objects with a structure specified by the parameter S which could be a set, bag, array, or another data type constructor, and no user-defined identifier (because they are self-naming objects). Class X can be defined using class T as its domain, and can have a membership constraint represented by C. All of the above structural semantics of a lexical object type X is captured by using a Membership macro with various parameter specifications. Similarly, the structural semantics of a non-lexical object type (NOLOT) X as shown in Nc.2 is captured in a Membership macro with different parameter values. The connection between a LOT and a NOLOT shown in Nc.3 is called a 'bridge' in NIAM and is mapped into an 'association' in ORECOM, whose semantic property is captured by four macros as given in Figure 5.2(a). The two Participation macros describe

that both classes X and Y are partially participated in this X-Y association. (The Participation macro was explained in Section 3.3.) The first Cardinality macro specifies that each X object *can* associate with at least one and at most one (i.e., exactly one) Y object. The same macro is used again for Y objects so that each Y object *can* associate with at least one and at most all (i.e., any number of) X objects. Nt.3.1 is the same as Nc.3 except that a 'role-total' constraint is added to the LOT side so that Y objects are totally participated in this X-Y bridge. This constraint leads to a different second Participation macro for Nt.3.1, which specifies the total participation of Y. The macro representations of NIAM and IDEF-1X are partially shown in Figure 5.2(a) and (b). A complete version of these representations is stored in the subsystem-1 for equivalence analysis.

Nc.1 (X)	=	Ic.1 (X)
Nc.2 (X)	=	Ic.2 (X)
Nc.3 (X, Y)	=	Ic.3 (X, Y)
Nt.3.1 (X, Y)	=	Ic.3 (X, Y) + Participation(Y, all, all, X)
		•
		•

Figure 5.3. The Equivalence Matrix for translating NIAM to IDEF-1X.

In the last step (i.e., equivalence analysis), the Nc.1, Nc.2, Nc.3, and Nt3.1 of NIAM are examined to find their equivalent construct and constraint in IDEF-1X based on their macro representations. It can be seen from the figure that Nc1, Nc.2, and Nc.3 are completely equivalent to Ic.1, Ic.2, and Ic.3, respectively, because of their identical macro representations. For Nt.3.1, however, no equivalent macro representation can be found in IDEF-1X. The closest one is Ic.3 which needs an adjustment on its second macro in order to make these two constructs equivalent. The result is "Nt.3.1 = Ic.3 + Participation(Y, all, all, X)", where the Participation macro (showing the total participation of Y) will automatically override the original one in Ic.3 (showing the partial participation of Y). The

output of the equivalence analysis for Nc.1, Nc.2, Nc.3, and Nt.3.1 is shown in Figure 5.3 and is stored in the Equivalence Matrices of subsystem-1.

5.2 Schema Translation

We start the schema translation in subsystem-2 by a schema compilation which is followed by an equivalence searching and a schema generation. The source schema given in Figure 5.1(a) is first compiled and its output is a list of constructs and constraints of NIAM as shown in Figure 5.4(a). This list is then passed to the Equivalence Search Module, where a table lookup is performed on the equivalence matrix in Figure 5.3 to find the equivalent IDEF-1X representations of Nc.1, Nc.2, and Nt.3.1. The search result in IDEF-1X is shown in Figure 5.4(b).

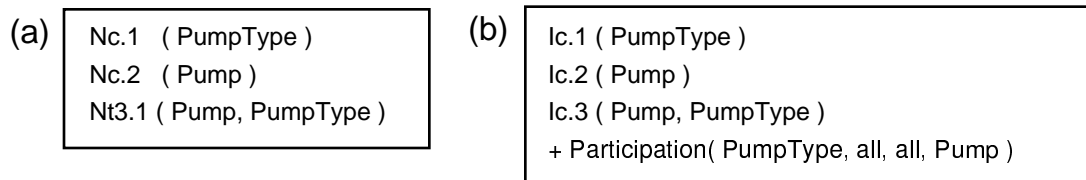


Figure 5.4 (a) The compiled source schema of NIAM.
(b) The equivalent constructs and constraints of IDEF-1X.

Finally, the Schema Generator generates the target schema, which is shown in Figure 5.1(b), based on these IDEF-1X constructs and constraints. Note that the target schema also includes a Participation macro so that its semantics can be completely equivalent to the original source schema shown in Figure 5.1(a). This added macro (i.e., the discrepancy) can be explicitly expressed in the form of micro-rules and be used to generate an explanation of the discrepancy or some program codes by a Rule Translator. When a user develops his/her applications based on the generated schema, the missing constraint of the IDEF-1X schema diagram can be compensated by including the constraint in the application programs.

6. Conclusions

In this paper we have proposed an approach of using ORECOM as the neutral core model to facilitate the rule-based data model and schema translation. An attempt has been made to identify the primitive concepts of this core model, including object, class, association, object operation, micro-rule, and macro (or constraint type). The distinguishing features of using ORECOM for data model and schema translation are summarized as follows.

- (1) The object orientation of ORECOM allows complex physical objects, abstract things, events, processes, and functions, etc., which are modeled by different constructs in different data models (e.g., entities, tuples, relationships, categories, etc.), to be uniformly modeled as objects and associations among objects. Thus, ORECOM is neutral in the sense that the basic structural properties of the existing high-level data models can be uniformly represented by ORECOM's neutral and low-level structural constructs.
- (2) ORECOM has a very powerful rule specification component which allows the various behavioral constraints (e.g., primary keys, cardinality mappings, total participations, derived attributes, etc.) of the high-level data models to be unambiguously defined by macros and micro-rules. These macros and micro-rules are used as the basic units to relate the semantic similarities and differences of the modeling constructs and constraints of high-level data models.
- (3) The ORECOM-based translation system is extensible. Therefore, should the existing models be extended or enhanced, or should new data models be introduced, the extensible core model and therefore the translation system can be extended to accommodate these changes and additions.
- (4) The translation algorithm, the strategy and the overall framework of the translation system is general. Since all the high-level data models are decomposed into the uniform ORECOM representations, only one general equivalence analysis module and one

general search module are required to perform the generation and the search of equivalence information for each pair of data models. Therefore, specific translation algorithms which are needed in pair-wise direct translation are avoided.

The important issue of semantic preserving translation has been addressed. The techniques for data model decomposition and equivalence analysis, which are the two key components of the translation system, based on the ORECOM model have been proposed and described in detail. We have also presented a simple example of NIAM to IDEF-1X schema translation to illustrate the methodology and architecture of the data model and the schema translation system. Some possible applications of this system such as schema sharing and database conversion as well as its impact on schema integration and schema design have also been discussed. We believe that the translation methodology and the implemented schema translation system presented in this paper are useful for achieving data sharing and interoperability of heterogeneous computing systems.

REFERENCES

- [ALON91] R. Alonso, D. Barbara, S. Cohn, "Data sharing in a large heterogeneous environment," IEEE Proceedings of the 7th International Conference on Data Engineering, 1991.
- [BARS92] T. Barsalou, D. Gangopadhyay, "M(DM): An open framework for interoperation of multimodel multidatabase systems," IEEE Proceedings of the 8th International Conference on Data Engineering, Feb. 1992.
- [BENN92] W. Benn, C. Kortenbreer, G. Schlageter, X. Wu, "On interoperability for KBMS Applications - The Horizontal Integration Task-," IEEE Proceedings of the 8th International Conference on Data Engineering, Feb. 1992.
- [BILL79] H. Biller, "On the equivalence of data base schemas -- A semantic approach to data translation," Information systems, vol. 4, no. 1, 1979.
- [BREI86] Y. Breitbart, P. L. Olson and G. R. Thompson, "Database integration in a heterogeneous database system," IEEE Proc. International Conference on Data Engineering, 1986.
- [BRIA85] H. Briand, H. Habrias, J. F. Hue and Y. Simon, "Expert system for translating an E-R diagram into databases," IEEE Proc. international conf. on E-R approach, 1985.
- [CARD80a] A. F. Cardenas and M. H. Pirahesh, "Data base communication in a heterogeneous data base management system network," Information Systems, vol. 5, no. 1, 1980.
- [CARD80b] A. F. Cardenas and M. H. Pirahesh, "The E-R model in a heterogeneous data base management system network architecture," Entity-Relationship Approach to System Analysis and Design, P. P. Chen (ed.), North-Holland, 1980.
- [CARD87] A. F. Cardenas, "Heterogeneous distributed database management : The HD-DBMS," Proc. of the IEEE, vol. 75, no. 5, May 1987.
- [CHEN76] P. P. Chen, "The Entity-Relationship model - toward a unified view of data," ACM Transactions on Database Systems, vol. 1, no. 1, Mar. 1976.
- [DEEN81] S. M. Deen, D. Nikodem and A. Vashishta, "The design of a canonical database system (PRECI)," The Computer Journal, vol. 24, no. 3, 1981.

- [DEMU87] S. A. Demurjian "The multi-lingual database system -- A paradigm and test-bed for the investigation of data-model transformations, data-language translations and data-model semantics," Ph.D. dissertation, Ohio State Univ., Columbus, Mar. 1987.
- [DEMU88] S. A. Demurjian, D. K. Hsiao "Towards a better understanding of data models through the multilingual database system," IEEE transactions on software engineering, vol. 14, no. 7, Jul. 1988.
- [DEVO80] C. Devor, J. Weeldreyer, "DDTS: A testbed for distributed database research," Proc. ACM Pacific '80 Conference, San Francisco, CA, Nov. 1980.
- [ELMA89] R. Elmasri, S. B. Navathe, Fundamentals of Database Systems, Chapter 12, 15, The Benjamin/Cummings Publishing Co., Inc., 1989.
- [FISH87] D. Fishman, et al., "Iris: An object-oriented database management system," ACM Transactions on Office Information Systems, vol. 5, no. 1, 1987.
- [FLYN85] D. J. Flynn and A. H. F. Laender, "Mapping from a conceptual schema to a target internal schema," The Computer Journal, vol. 28, no. 5, 1985.
- [GANG91] D. Gangopadhyay, T. Barsalou, "On the semantic equivalence of heterogeneous representations in multimodel multidatabase systems," ACM SIGMOD RECORD, Vol. 20, No. 4, Dec. 1991.
- [GRAN84] J. Grant, "Constraint preserving and lossless database transformations," Information Systems, vol. 9, no. 2, 1984.
- [HEIM85] D. Heimbigner, D. McLeod, "A federated architecture for information systems," ACM Transactions on Office Information Systems, Vol. 3, No. 3, Jul, 1985.
- [HSIA89] D. K. Hsiao, M. N. Kamel, "Heterogeneous Databases: proliferations, issues, and solutions," IEEE Transactions on knowledge and data engineering, vol. 1, no. 1, Mar, 1989.
- [HULL87] R. Hull, and R. King, "Semantic Database Modeling: Survey, Application, and Research Issues," ACM Computing Surveys, 19(3), Sept. 1987, pp. 201-258.
- [HWAN83] H. Y. Hwang and U. Dayal "Using the Entity-Relationship model for implementing multi-model database systems," Entity-Relationship Approach to Information Modeling and Analysis, P. P. Chen (ed.), North-Holland, 1983.
- [IISS86] "The integrated information support system," Gateway, vol. 2, no. 2, Industrial Technology Institute, Mar.-Apr. 1986.
- [JACO85] B. E. Jacobs, Applied Database Logic, Vol. I, Fundamental Database Issues, Prentice Hall, 1985.
- [KAME92] N. Kamel, P. Wu, S. Y. W. Su, "Pattern based object calculus," submitted to The International Journal of Very Large Data Bases.
- [KATZ80] R. H. Katz "Database design and translation for multiple data models," Ph.D. dissertation, University of California, Berkeley, Jun. 1980.
- [KENT91] W. Kent, "The breakdown of the information model in multi-database systems," ACM SIGMOD RECORD, Vol. 20, No. 4, Dec. 1991.
- [KRIS88] V. Krishnamurthy, S. Y. W. Su, H. Lam, M. Mitchell and E. Barkmeyer "IMDAS -- An integrated manufacturing data administration system," Data & knowledge engineering, North-Holland, 1988.
- [KLUG78] A. C. Klug, "Theory of database mappings," Ph. D. dissertation, University of Toronto, Canada, 1978.
- [KRIS91] R. Krishnamurthy, W. Litwin, W. Kent, "Language features for interoperability of databases with semantic discrepancies," ACM SIGMOD Proceedings of the International Conference on Management of Data, 1991.
- [LARS83] J. A. Larson, "Bridging the gap between network and relational database management systems," IEEE Computer, September, 1983.
- [LEUN88] C. M. R. Leung and G. M. Nijssen, "Relational database design using the NIAM conceptual schema," Information systems, vol. 13, no. 2, 1988.
- [LITW86] W. Litwin, A. Abdellatif, "Multidatabase interoperability," IEEE Computer, Dec. 1986.
- [LOOM86] M. E. S. Loomis, "Data modeling -- The IDEF-IX technique," IEEE communications, Mar. 1986.
- [LYNG87] P. Lyngbaek and V. Vianu, "Mapping a semantic database model to the relational model," Proc. of ACM SIGMOD, 1987.
- [MORG83] M. Morgenstern, "A unifying approach for conceptual schema to support multiple data models," Entity-Relationship approach to information modeling and analysis, P. P. Chen (ed.), North-Holland, 1983.
- [NSF89] Proceedings of the 1989 NSF Workshop on Heterogeneous Databases, Dec. 1989.
- [OZKA90] E. Ozkarahan, Database Management Concepts, Design, and Practice, Chapter 10, Prentice Hall, 1990.
- [PECK88] J. Peckham, F. Maryanski, "Semantic data models," ACM Computing Surveys, vol. 20, no. 3, Sep. 1988.
- [POTT84] W. D. Potter "DESIGN-PRO : A multi-model schema design tool in PROLOG," Proc. first international workshop on expert database systems, 1984.

- [SEN86] A. Sen, and C. Ching, "Schema translation : A three-level semantic abstraction approach," Information systems, vol. 11, no. 3, 1986.
- [SCHE89] D. A. Schenck, "Information modeling language : EXPRESS," Language reference manual, ISO TC184/SC4/WG1, N362, May, 1989.
- [SHET90] A. P. Sheth, J. A. Larson, "Federated database systems for managing distributed heterogeneous, and autonomous databases," ACM Computing Surveys, Vol. 22, No. 3, 1990.
- [SMIT81] J. M. Smith et al. "Multibase -- integrating heterogeneous distributed database systems," Proc. of the National Computer Conference, 1981.
- [SU89] S. Y. W. Su, V. Krishnamurthy, and H. Lam, "An object-oriented semantic association model (OSAM*)," in AI in Industrial Engineering and Manufacturing : Theoretical Issues and Applications, Kumara, S., et. al., (ed), American Institute of Industrial Engineering, 1989.
- [SU91] S. Y. W. Su and A. M. Alashqur, "A pattern based constraint specification language for object oriented databases," Proc. IEEE COMPCON, 1991.
- [URBA91] S. D. Urban, J. Wu, "Resolving semantic heterogeneity through the explicit representation of data model semantics," ACM SIGMOD RECORD, Vol. 20, No. 4, Dec. 1991.
- [VASS80] Y. Vassiliou, F. H. Lochovsky, "DBMS transaction translation," IEEE Proc. of Computer Software and Applications Conference, Oct. 1980.
- [VERH82] G. M. A. Verheijen, and J. V. Bekkum, "NIAM : An information analysis method," on Information System Design Methodologies : A Cmparative Review, Olle, T. W. et. al., (ed), North-Holland, 1982.