Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Management and Translation of Heterogeneous Database Schemes in a Lattice Framework

Paolo Atzeni, Riccardo Torlone

RT-INF-27-97 **Giugno 1997**

This work was partially supported by MURST and by Consiglio Nazionale delle Ricerche.

ABSTRACT

In this paper we study the problem of translating schemes between different data models, in a formal framework that refers to a wide range of models. We first introduce a graph-theoretic formalism that allows us to uniformly represent schemes and models, to compare different data models and to describe the behavior of basic translations. The formalism is based on a classification of the constructs used in the known data model into a limited set of types. Then, we study in this framework formal properties of scheme translation between heterogeneous data models, and we develop a general methodology for deriving translations that enjoy those properties.

Keywords: Data model, metamodel, heterogeneity, interoperability, schema translation, graphical representation, lattice structure.

1 Introduction

1.1 Motivations and background

It is widely accepted now that a conceptual data model be used in the analysis phase and many tools exist that support the analysis and design of information systems (see for instance the book of Batini et al. [8]). At the same time, many data models have been defined (see the survey of Hull and King [15], and the book of Tsichritzis and Lochovski [22]), and each of the tools usually implements only one of the models. Incidentally, although it is true that most of the tools support the Entity-Relationship model, introduced by Chen [12], it is also true that there are in fact many versions of this model, which represent actually different models, often compatible only to a limited extent. It is reasonable that in a complex environment different models should coexist, for a number of reasons: (i) different subproblems may be analyzed independently (for examples by companies that later merged or got involved in a federated project) or (ii) different analysts may prefer different models; or (iii) different subproblems could be tackled with different models, because of the specific aspects of each.

We have already discussed the general ideas of our approach [3]. Our long term goal is an environment that allows the definition of "any reasonable model," by means of a suitable formalism called a *metamodel*. Then, for any two models M_1 and M_2 defined in this way, and for each scheme S_1 (the *source scheme*) of M_1 (the *source model*), it should be possible to obtain a scheme S_2 (the *target scheme*) that be the translation of S_1 into M_2 (the *target model*). A major point in this plan is related to the expressive power of the metamodel, that is, the set of models that can be defined. In fact, the notion of a model is widely accepted and understood, but there is no general, formal definition. This problem can be overcome by noting that, according to a classification of Hull and King [15], all the constructs used in most known models fall in a rather limited set of categories: lexical type, abstract type, aggregation, generalization, function, grouping. Therefore, we have argued (see [3]) that a metamodel can be defined by means of a basic set of *metaconstructs*, corresponding to the above categories. Then, a model can be described by defining its constructs by means of the metaconstructs in the metamodel. The generality and variety of the metaconstructs determine the expressive power of the metamodel. In a sense, this approach is "asymptotically" complete: if there is a model that cannot be expressed by means of the metamodel, because a construct in the model has no counterpart in the metamodel, then the metamodel can be extended by introducing a new, suitable metaconstruct. We believe that a wide class of models can be managed with a few extensions to the basic metamodel.

A second key point in our approach is that there is no clear notion of when a translation is correct: at first, one could think that the target scheme should be "equivalent" to the source scheme, that is, they should represent "the same information." A lot of research has been conducted in the last decades on scheme equivalence (or "comparison of information capacity") with reference to the relational model [2, 14, 20] or to heterogeneous frameworks [1, 16, 18, 19], but there is no general, agreed definition.

To the best of our knowledge, there is not much literature related to the problem we tackle and the goal we set. Some work exists on the idea of a metamodel for the representation of models [7, 16], but the goal is more on the integration of heterogeneous databases in a federated environment (Sheth and Larson [21] provided a survey on

federated database systems) than on the translation of schemes to generic target models.

On the basis of the above arguments, one of the conclusions of our preliminary study was that in order to study transformations between different data models, a pragmatic approach is needed. As a matter of fact, the various constructs that correspond to the same metaconstruct can be assumed to have the same semantics (at least with respect to the translation process). For example, as argued by Hull and King in [15], the semantics of entities is the same in all versions of the E-R model, as well as in the functional model and corresponds to the semantics of abstracts (or non-printable) types. As a consequence, the process of scheme translation can be based on translations of constructs (or simple combinations thereof) defined with respect to the corresponding metaconstruct. In this way, schemes and translations need not refer to specific models. For example, a scheme S_1 with entities, binary relationships and is-a hierarchies belongs to all versions of the E-R model that allow is-a hierarchies, whereas a scheme S_2 with binary and ternary relationships and no hierarchies belongs to all versions that allow ternary relationships. The translation of S_1 into the model of S_2 would be performed within the framework of a model that allows both ternary relationships and is-a hierarchies, but its result would not include is-a hierarchies.

Another result of our preliminary study was that the translation process can be obtained as the composition of a predefined set of elementary transformations which implement the standard translations between constructs studied in the literature (see [9]). The operation that eliminates is-a hierarchies (and, say, replaces them with binary relationships) is an example of basic translation that should be used in many translations that go from a model with hierarchies to a model without them.

1.2 Contributions of the paper

The approach has been studied within a graph-theoretic framework that allows us to define in a uniform way schemes and models. We have the notion of a *structure*, a directed graph whose nodes have different types (corresponding to the basic metaconstructs we mentioned above, such as lexical, abstract, aggregation and function). Then, we use structures in two ways: we have *patterns*, structures where the edges have cardinalities as labels, and *schemes*, structures whose nodes and edges are labeled with names. Then a model M is defined by means of a set of patterns $\mathcal{P}$: a scheme S *is allowed* in M if S can be mapped (according to a specific, but natural notion) to one of the patterns in $\mathcal{P}$. In this way, there is a set of schemes associated with each model. Clearly, a scheme can be allowed in several models. It turns out that a partial order can be introduced on patterns, which, suitably extended, becomes a lattice on sets of patterns.

The main goal of this paper is to study within this framework how elementary transformations can be composed, in order to form complex translations: assuming that the basic transformations are correct, we first introduce correctness and other desirable properties of complex translations, and then we present techniques for finding translations that satisfy those properties. In a sense, this could be called an "axiomatic" approach that is coherent with the observation made above on the difficulty in defining correct translations. Now, elementary transformations are described on the basis of the patterns they eliminate and the patterns they introduce: clearly, this is only part of their description (we say this is the *signature* of a translation step, as opposed to its *body* or *program*), but it is sufficient for our purposes. The main point of the approach is that if a transforma-

tion eliminates all constructs of a scheme that correspond to a pattern P_1 and replaces constructs corresponding to a pattern P_2, then, from an intuitive point of view, its application to a set of schemes described by a set of patterns $\mathcal{P}$ generates schemes described by another set of patterns (intuitively, $\mathcal{P} - \{P_1\} \cup \{P_2\}$. In this way, a translation from a model M_0 (described by the set of patterns $\mathcal{P}_0$) to another model M (described by $\mathcal{P}$) can be seen as a sequence of elementary translations $\tau_1, \ldots, \tau_k$ such that there is a sequence of sets of patterns $\mathcal{P}_1, \ldots, \mathcal{P}_k$ such that $\mathcal{P}_k = \mathcal{P}$ and τ_i applied to schemes described by $\mathcal{P}_{i-1}$ generates schemes described by $\mathcal{P}_i$.

Thus, the main contributions of this paper are the definition and characterization of desirable properties of translations, and the development of a methodology for the automatic generation of translations that satisfy such properties. The results are obtained in an elegant way by means of the lattice framework on patterns.

1.3 Plan of the paper

The paper is organized as follows. In Section 2 we describe our graph-theoretic formalism for the various components of the metamodel, and show its properties. This formalism is used in Section 3 to define schemes, models and translations of schemes. In Section 4, we study formal properties on the activity of scheme translation and provide several characterizations. On the basis of these results, in Section 5 we develop a general method to derive translations between models. In Section 6, we show a detailed example to illustrate the various notions introduced throughout the paper. Finally, in Section 7, we sketch some conclusions. Because of space limitation, the proofs of the various results of this paper are not given here.

2 A graph-theoretic formalism

2.1 Graphs and trees

Before introducing our graph-theoretic formalism, we fix some terminology from graph theory. A directed graph (in this paper we only consider directed graphs and so we will often omit the adjective directed) is a pair $G = (V, E)$ where V is a finite set of *nodes* (or *vertices*) and E is a set of ordered pairs of nodes called *edges*. Given an edge $e = (v_i, v_j)$, we say that v_i and v_j are the *tail* and the *head* of e, respectively, and that e is *outgoing* from v_i and *incoming* in v_j.

A *path* in a directed graph $G = (V, E)$ is a sequence of nodes $v_1, \ldots, v_k$ in V, $k > 0$, such that there is edge (v_i, v_{i+1}) in E, for each $i, 1 \leq i < k$. We say that the path is *from* v_1 *to* v_k. Given a directed graph $G = (V, E)$ and a subset V' of V, the *subgraph of* G *induced by* V' is the graph $G' = (V', E')$ where E' is the subset of E whose edges have both head and tail in V'. Given a graph G and a node v of G, the *subgraph of* G *generated by* v is the subgraph of G induced by v and all the nodes v' of G such that there is a path from v to v'.

A (directed) *tree* is a directed graph with the following properties. There is one node, called *root*, without incoming edges and from which there is a path to every node and each node other than the root has exactly one incoming edge.

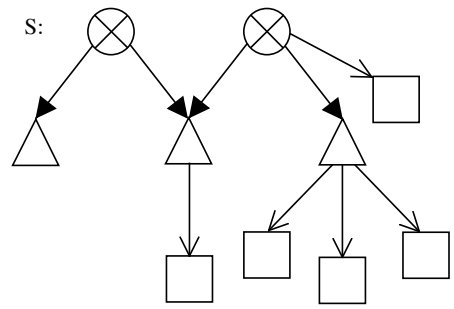


Figure 1: An example of a structure.

2.2 Structures and Patterns

Let us now introduce the graphical framework for describing models and schemes. We fix a set of *node types* $\mathcal{N}$ and the set of *edge types* $\mathcal{E}$: they will be used to represent the metaconstructs of the metamodel. Specifically, following a notation used by Hull and King in [15], we use nodes to represent basic (or atomic) data types and type constructors and edges to represent functions (or attributes) and applications of type constructors to other data types. In our examples, we will consider a metamodel involving three types of nodes corresponding to *abstracts* (denoted by the symbol $\triangle$), *aggregations* ($\otimes$), and *lexicals* ($\square$); and six types of edges corresponding to *functions* (denoted by $\rightarrow$), *multivalued functions* ($\twoheadrightarrow$), *components of aggregation* ($\longrightarrow\!\!\triangleright$), *key of aggregation* ($-\!\!\bullet\!\!-\!\!\triangleright$), *key of abstract* ($-\!\!\bullet\!\!\rightarrow$) and *subset relations between abstracts* ($\Rightarrow$). We point out however that the approach can handle a variety of metamodels [3]: new types of nodes and edges can be included to capture specific constructs without affecting the general results that follow.

Definition 2.1 *A* structure *is a triple $S = (G, \mu, \epsilon)$ where $G = (V, E)$ is a directed acyclic graph, and μ and ϵ are* structuring functions*: $\mu : V \rightarrow \mathcal{N}$ and $\epsilon : E \rightarrow \mathcal{E}$.*

Example 2.1 *Figure 1 shows a simple structure. The structure has two aggregations over abstracts and functions from abstracts to lexicals.*

We assume that every structure satisfies a number of conditions corresponding to the usual restrictions on the composition of constructs. Note that for sake of simplification, we have assumed that a structure has no directed cycles. This is not a significant limitation in practice: intuitively, "cyclic" schemes can be obtained by composing structures, in the same way as in conceptual models cycles in the schemes appear, without recursive constructs (a formal justification for this argument was given by Kuper in [17]).

Given two structures $S_1 = ((V_1, E_1), \mu_1, \epsilon_1)$ and $S_2 = ((V_2, E_2), \mu_2, \epsilon_2)$, a *structure-preserving* mapping Φ from S_1 to S_2 is a pair of functions $\theta : V_1 \rightarrow V_2$ and $\phi : E_1 \rightarrow E_2$ such that: (1) for each $v \in V_1$, $\mu_1(v) = \mu_2(\theta(v))$, and (2) for each $e = (v, v') \in E_1$, $\epsilon_1(e) = \epsilon_2(\phi(e))$ and $\phi(e) = (\theta(v), \theta(v'))$. Thus, a structure-preserving mapping preserves the type of nodes and edge, and the "topology" of the structure. If θ and ϕ are bijections then we say that Φ is an *isomorphism* and that S_1 and S_2 are *isomorphic* (note that two finite isomorphic structures are indeed identical).

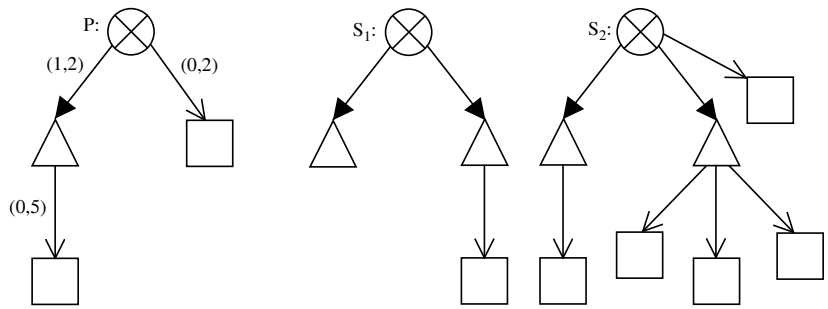


Figure 2: A pattern and two tree structures that match with it.

Let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{N}^+$ the set of positive natural numbers. A *range of cardinality* (or simply, a *range*) is a pair (n, m) (*minimum* and *maximum* cardinality respectively) such that $n \in \mathbb{N}$, $m \in \mathbb{N}^+$ and $n \leq m$. If $k \in \mathbb{N}$ and $r = (n, m)$ is a range, then k is in r, in symbol $k \in r$, if $n \leq k \leq m$.

Definition 2.2 *A* pattern *is a pair $P = (S, \rho)$ where $S = (G, \mu, \epsilon)$ is a structure such that G is a tree, and ρ is a function that associates a range with each edge of G.*

Intuitively, a range denotes the minimum and the maximum number of edges of a certain type that can outgo from a specific node in a structure. Thus, a pattern describes a specific composition of metaconstructs with some degree of freedom on the number of edges ougoing from a node, and therefore is a representative of an entire class of structures.

Example 2.2 *Figure 2 shows a pattern (P) and two tree structures (S_1 and S_2). The pattern P represents unary and binary aggregations of abstracts with optional functions from abstracts to lexicals (with a maximum of five) and optional functions from aggregations to lexicals (maximum two). It easily follows that P is a representative of both S_1 and S_2.*

For sake of conciseness, in our examples we will also represent patterns by means of a parenthetical notation (this is always possible since a pattern is a tree). Specifically: $v_1 \rightarrow v_2$ will be denoted by $v_1\langle v_2\rangle$, $v_1 \twoheadrightarrow v_2$ by $v_1\langle\langle v_2\rangle\rangle$, $v_1 \longrightarrow\!\!\triangleright v_2$ by $v_1[v_2]$, $v_1 \;-\!\!\bullet\!\!\rightarrow\!\triangleright\; v_2$ by $v_1[v_2]^k$, $v_1 -\!\!\bullet\!\!\rightarrow v_2$ by $v_1\langle v_2\rangle^k$, and $v_1 \Rightarrow v_2$ by $v_1[\![v_2]\!]$.

The main definition of this section establishes a precise correspondence between structures and sets of patterns. We first need to introduce some preliminary notions. We say that a node v in a pattern is *mandatory* if it is the root or all the edges in the path from the root to v have a minimum cardinality greater than 0. Then, we say that a tree structure S (that is, a structure whose graph is a tree) *matches* with a pattern $P = (S', \rho)$ if there is a structure-preserving mapping $\Phi = (\theta, \phi)$ from S to S' that verifies the following conditions: (1) θ is surjective on the mandatory nodes of P, and (2) for each node v of S, the number k of edges e of S, outgoing from v, such that $\phi(e) = e'$, for some edge e' of S' is in the range of e', that is, $k \in \rho(e')$.

Example 2.3 *The pattern P in Figure 2 has two mandatory nodes: the aggregation and the abstract. It is easy to see that the two tree structures S_1 and S_2 match with P according to the definition above.*

Now, given a generic structure S whose graph is not a tree, we can derive in a unique way a number of trees from it that we call *components* of S. These components can be obtained by simply applying the following rule exhaustively, until no modification can be performed:

- if a node v has several incoming edges, we attach to each of them a copy of the subgraph induced by v.

Definition 2.3 *A structure S is an* instance *of a set of patterns $\mathcal{P}$ if, for each* component *S_i of S there is a pattern $P \in \mathcal{P}$ such that S_i matches with P.*

Given a set of patterns $\mathcal{P}$, we will denote with $Inst(\mathcal{P})$ the set of all instances of $\mathcal{P}$.

Example 2.4 *The structure S in Figure 1 has two components, which indeed correspond to the structures S_1 and S_2 in Figure 2. It follows that S is an instance of the pattern P in Figure 2. A more involved example is reported in Figure 3 that shows a set of patterns representing a version of the E-R model (in this figure n is a parameter denoting a fixed integer). The instances of this set of patterns may be: (1) isolated abstracts with key combined with (monovalued or multivalued) functions from abstracts to lexicals (pattern P_1), and (2) many-to-many binary aggregations of abstracts combined as above (pattern P_2), and (3) one-to-many binary aggregations of abstracts combined as above (pattern P_3). An example of instance of this set of patterns is reported in Figure 4.*

Definition 2.4 *A set of patterns $\mathcal{P}_1$* is dominated *by a set of patterns $\mathcal{P}_2$, in symbols $\mathcal{P}_1 \subseteq \mathcal{P}_2$, if $Inst(\mathcal{P}_1) \subseteq Inst(\mathcal{P}_2)$. If both $\mathcal{P}_1 \subseteq \mathcal{P}_2$ and $\mathcal{P}_2 \subseteq \mathcal{P}_1$ then we say that the two sets of patterns have* the same representation capacity *(in symbols, $\mathcal{P}_1 \equiv \mathcal{P}_2$).*

Example 2.5 *It is easy to see that the set of patterns in Figure 5 dominates (and therefore describes at least all the structures described by) the set of patterns in Figure 3.*

2.3 A partial order on patterns

We now define a partial order relationship between them that yields a practical way to compare their representation capacities. It refers to patterns of a particular kind: we say that a pattern P is *unitary* if all the edges of P have range $(1, 1)$. It turns out that by using a set of *decomposition rules*, it is always possible to transform (in a unique way) a set of patterns $\mathcal{P}$ into a set of unitary patterns $\mathcal{P}^*$ having the same representation capacity as $\mathcal{P}$. Specifically, the *decomposition* of $\mathcal{P}$, denoted with $\mathcal{P}^*$, is the set of patterns obtained by applying the following rules exhaustively, until no modification can be performed:

- if there is a pattern $P = (S, \rho)$ in $\mathcal{P}$ with an edge $e = (v_1, v_2)$ such that $\rho(e) = (0, y)$ then P is replaced by two patterns P_1 and P_2 such that: (1) $P_1 = (S, \rho_1)$ where $\rho_1 = \rho$ except that $\rho_1(e) = (1, y)$, and *(ii)* $P_2 = (S_2, \rho_2)$ where S_2 is obtained from S by deleting e and the tree generated by v_2, and ρ_2 is the restriction of ρ to S_2;

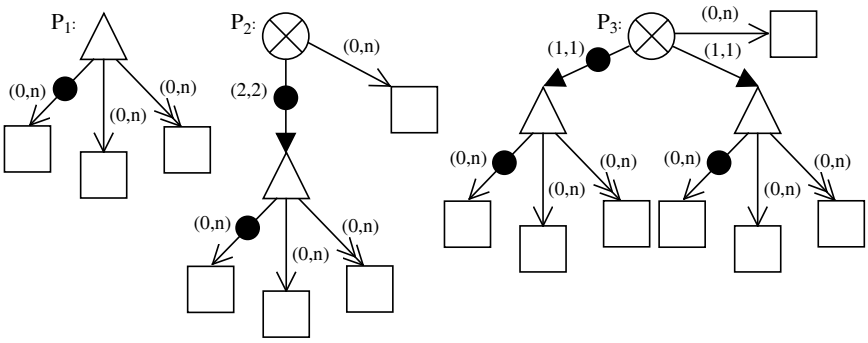


Figure 3: A set of patterns describing a version of the E-R model.

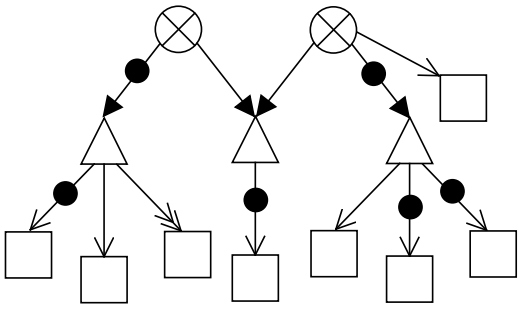


Figure 4: A structure that is an instance of the set of patterns in Figure 3.

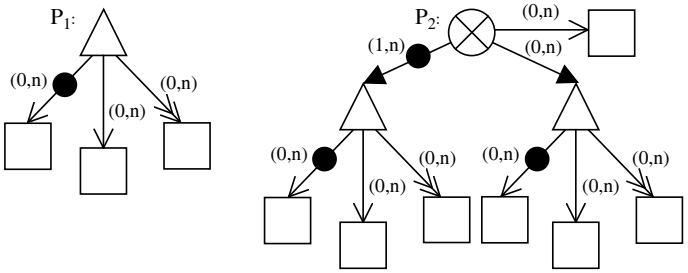


Figure 5: A set of patterns that dominates the set of patterns in Figure 3.

- if there is a pattern $P = (S, \rho)$ in $\mathcal{P}$ with an edge $e = (v_1, v_2)$ such that $\rho(e) = (x, y)$ and $x > 0$ then P is replaced by $y - x + 1$ patterns $P_0, P_1, \ldots, P_{y-x}$ such that each P_i, $i = 0, \ldots, y - x$, is obtained from P by replacing e with $x + i$ edges e_i with $\rho(e_i) = (1, 1)$ and attaching to the tails each e_i a copy of the subtree of P generated by v_2.

Lemma 2.1 *For every set of patterns $\mathcal{P}$, the decomposition of $\mathcal{P}$ (1) is unique, (2) contains only unitary patterns, and (3) has the same representation capacity as $\mathcal{P}$.*

A partial order relationship on sets of patterns is then defined as follows. We say that two patterns $P_1 = (S_1, \rho_1)$ and $P_2 = (S_2, \rho_2)$ are *isomorphic* if S_1 and S_2 are isomorphic and each edge of P_1 is associated with an edge of P_2 with the same range.

Definition 2.5 *A set of patterns $\mathcal{P}_1$ is* subsumed *by a set of patterns $\mathcal{P}_2$ (in symbol $\mathcal{P}_1 \preceq \mathcal{P}_2$) if for each pattern $P_1 \in \mathcal{P}_1^*$ there is a pattern $P_2 \in \mathcal{P}_2^*$ such that P_1 and P_2 are isomorphic. If both $\mathcal{P}_1 \preceq \mathcal{P}_2$ and $\mathcal{P}_2 \preceq \mathcal{P}_1$ then $\mathcal{P}_1$ and $\mathcal{P}_2$ are* equivalent *(in symbols $\mathcal{P}_1 \sim \mathcal{P}_2$).*

Given a pair of sets of patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, the subsumption relationship yields a practical way for testing whether $\mathcal{P}_1$ is dominated by $\mathcal{P}_2$ (that is, whether $\mathcal{P}_2$ is able to represent all the structures represented by $\mathcal{P}_1$ and possibly more). This is confirmed by the following result.

Theorem 2.1 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two sets of patterns. Then $\mathcal{P}_1 \preceq \mathcal{P}_2$ if and only if $\mathcal{P}_1 \subseteq \mathcal{P}_2$.*

2.4 An algebra of patterns

We now define a number of binary operations on sets of patterns, based on the notion of decomposition.

- The *join* of a pair of sets of patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \sqcup \mathcal{P}_2$, is the set of patterns $\mathcal{P}_1^* \cup \mathcal{P}_2^*$.
- The *meet* of a pair of sets of patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \sqcap \mathcal{P}_2$, is the set of patterns $\mathcal{P}_1^* \cap \mathcal{P}_2^*$.
- The *difference* of a pair of sets of patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 - \mathcal{P}_2$, is the set of patterns $\mathcal{P}_1^* - \mathcal{P}_2^*$.

Lemma 2.2 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be a pair of sets of patterns. Then: (1) $Inst(\mathcal{P}_1 \sqcup \mathcal{P}_2) = Inst(\mathcal{P}_1) \cup Inst(\mathcal{P}_2)$, (2) $Inst(\mathcal{P}_1 \sqcap \mathcal{P}_2) = Inst(\mathcal{P}_1) \cap Inst(\mathcal{P}_2)$, and (3) $Inst(\mathcal{P}_1 - \mathcal{P}_2) = Inst(\mathcal{P}_1) - Inst(\mathcal{P}_2)$.*

The join and the meet operators are both commutative and associative and so we can speak of join and meet of a finite *collection* of sets of patterns. The following result states that the partial order relation $\preceq$ induces a *lattice* on the set of sets of (unitary) patterns.

Theorem 2.2 *Every finite collection $\mathbf{P}$ of sets of patterns has both a* greatest lower bound *($glb(\mathbf{P})$) equal to the meet of the elements in $\mathbf{P}$, and a* least upper bound *($lub(\mathbf{P})$) equal to the join of the elements in $\mathbf{P}$.*

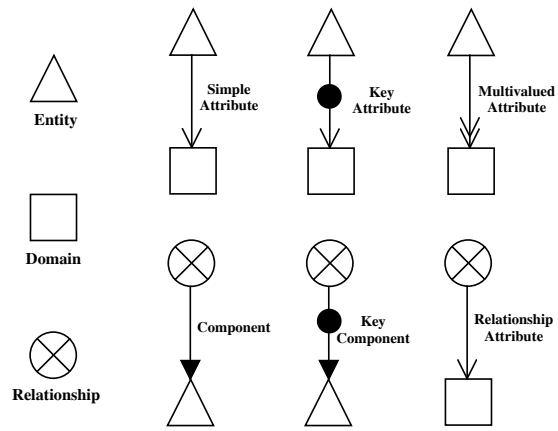


Figure 6: A labelling function for the set of patterns in Figure 3.

3 Models, Schemes and Translations

In this section we show how structures and patterns can be used to represent the various components of our framework.

3.1 Models and Schemes

Definition 3.1 *A* model *is a pair* $M = (\mathcal{P}, \gamma)$ *where* $\mathcal{P}$ *is a set of patterns and* γ *is a labeling function that maps each element of* $\mathcal{N} \cup \mathcal{E}$ *occurring in* $\mathcal{P}$ *to a label.*

The labels corresponds to the names associated with to a construct in a specific model (e.g., the abstract construct is called *entity* in the E-R model).

Example 3.1 *Figure 6 describes a labelling for the set of patterns in Figure 3, according to a quite standard terminology for the E-R model. Thus, the labelling function specify that the set of patterns indeed describe a version of the E-R model.*

Definition 3.2 *A* scheme *is a pair* $\mathbf{S} = (S, \lambda)$ *composed by a structure* S *and by a* labeling function λ *that maps each node and each edge of* S *to a label, such that different labels are associated with different elements of the structure.*

The labels used in a scheme correspond to names associated with the various concepts in a specific scheme (e.g., persons, books and so on). It is important to note that in our approach the definition of scheme is completely independent on the notion of model. Clearly, it is possible to establish a correspondence between schemes and models as follows.

Definition 3.3 *A scheme* $\mathbf{S} = (S, \lambda)$ *is* allowed *in a model* $M = (\mathcal{P}, \gamma)$ *if* $S \in Inst(\mathcal{P})$.

Example 3.2 *In Figure 7, we report a scheme, representing a real-world situation, that is allowed in the model described by the patterns in Figure 3.*

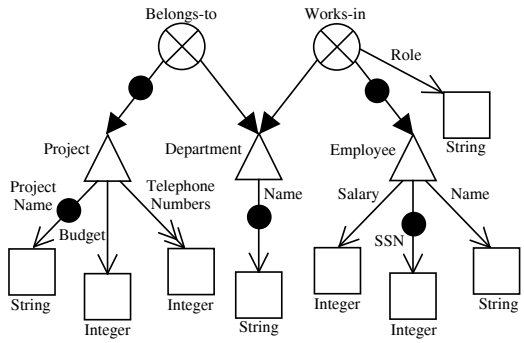


Figure 7: A scheme allowed in the model described in Figure 3.

Let S be a structure and $\Delta(S)$ be the set of patterns $P = (S_i, \rho)$ such that S_i is a component of S and $\rho(e) = (1, 1)$ for each edge e of S_i. The following result gives us an effective method to check whether a scheme belongs to a model.

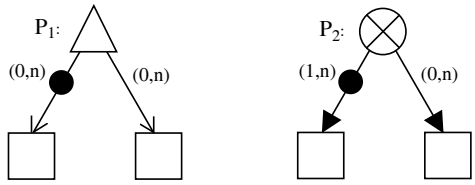
Lemma 3.1 *Let* $\mathbf{S} = (S, \lambda)$ *be a scheme and* $M = (\mathcal{P}, \gamma)$ *be a model. Then,* $\mathbf{S}$ *belongs to* M *if and only if* $\Delta(S) \preceq \mathcal{P}$.

3.2 Schema translations

A *schema translation* τ is a function that operates on structures by replacing constructs with other constructs. In our framework, translation functions can be specified by means of a specific language (a good candidate is the graph based database language introduced by Gyssen et al. in [13]). However, since the goal of this paper is to study general properties of translations between models, we need an higher level description of the behavior of a translation function. Again, this will be done by using the notion of pattern.

Definition 3.4 *A* translation signature σ *for a translation function* τ *is a pair of patterns* $\sigma = (P_1, P_2)$ *such that for each structure* S *and for each component* S_i *of* S*: (1)* $\tau(S_i)$ *is an instance of* $\{P_2\}$ *if* S_i *matches with* P_1*, and (2)* $\tau(S_i) = S_i$ *otherwise.*

Intuitively, a translation signature represents: (1) the constructs eliminated by τ and (2) the constructs introduced by τ, as effect of its execution. For instance, the following translation signature:



represents a translation that replaces abstracts and (optional) functions from abstracts to lexical (e.g., an entity of the E-R model with its attributes), with an aggregation on lexicals (e.g., a relation of the relational model). Note that, also in this case, a translation signature is independent of a specific model.

Definition 3.5 *A* translation rule *has the form $\sigma[\tau]$, where τ is a translation function and σ is a translation signature for τ. A* translation *T is a sequence of translation rules $T = \sigma_1[\tau_1], \ldots, \sigma_k[\tau_k]$.*

A nice property of a translation signature is that it can be used to characterize translations in terms of models. Let $\mathcal{P}$ be a set of patterns and $\sigma[\tau]$ be a translation rule where $\sigma = (P_1, P_2)$. Then, the *effect* of σ on $\mathcal{P}$, denoted by $\sigma(\mathcal{P})$, is defined as follows:

$$\sigma(\mathcal{P}) = \begin{cases} (\mathcal{P} - \{P_1\}) \sqcup \{P_2\} & \textit{if } \{P_1\} \preceq \mathcal{P} \\ \mathcal{P} & \textit{otherwise} \end{cases}$$

We have the following result.

Theorem 3.1 *Let $\sigma[\tau]$ be a translation rule, $\mathcal{P}$ a set of patterns and S a structure such that $S \in Inst(\mathcal{P})$. Then, it is the case that $\tau(S) \in Inst(\sigma(\mathcal{P}))$.*

In our approach, a complex translation T can be obtained as a composition of a number of predefined *basic* translation rules: $T = \sigma_1[\tau_1], \ldots, \sigma_k[\tau_k]$. These basic translations implement the standard translations between the constructs present in the traditional data models (e.g., from an entity of the Entity-Relationship model to a relation of the relational model, or from a n-ary relation to a set of binary ones). The effect of the execution of a complex translation T on a set of patterns $\mathcal{P}$, denoted by $\sigma_T(\mathcal{P})$, can be easily computed as the composition of the effects of the components of T.

More specifically, the *signature* σ_T of a translation $T = \sigma_1[\tau_1], \ldots, \sigma_k[\tau_k]$ is the sequence of signatures $\sigma_T = \sigma_1, \ldots, \sigma_k$, whereas the *body* τ_T of T is the composition of the translation functions of T: $\tau_T = \tau_k \circ \ldots \circ \tau_1$. Then, the effect of the signature $\sigma_T = \sigma_1, \ldots, \sigma_k$ over a set of patterns $\mathcal{P}$ is the set of patterns $\sigma_k(\ldots \sigma_1(\mathcal{P}) \ldots)$. It turns out that the results above can be easily extended to entire translations.

Corollary 3.1 *Let T be a translation (with signature σ_T and body τ_T), $\mathcal{P}$ be a set of patterns and S a structure such that $S \in Inst(\mathcal{P})$. Then, it is the case that $\tau_T(S) \in Inst(\sigma_T(\mathcal{P}))$.*

4 Properties of schema translations

According to our approach, we fix a set $\mathcal{R}_b$ of basic translation rules and we assume hereinafter that a translation T is *based on* $\mathcal{R}_b$, that is, a sequence of translation rules in $\mathcal{R}_b$. We recall that these basic translations implement the standard translations between the constructs present in the traditional data models (e.g., from an entity of the Entity-Relationship model to a relation of the relational model or from a n-ary relation to a set of binary ones) [3].

4.1 Correctness of translations

We now introduce a very general and natural notion of correctness of translations: we say that a translation T is a *correct translation from a source model M_s to a target model M_t* if the application of T to schemes allowed in M_s generates schemes allowed to M_t. By the results of the previous section it follows that this property can be easily verified by means of the translation signatures.

Lemma 4.1 *A translation T is a correct translation from a source model $M_s = (\mathcal{P}_s, \gamma_s)$ to a target model $M_t = (\mathcal{P}_t, \gamma_t)$ if and only if $\sigma_T(\mathcal{P}_s) \preceq \mathcal{P}_t$.*

The following important result follows by the lattice structure on the set of patterns.

Theorem 4.1 *Let $\mathcal{M} = \{M_1, \ldots, M_k\}$ be a set of models $M_i = (\mathcal{P}_i, \gamma_i)$ $(i = 1, \ldots, k)$, $\mathcal{P}_\top$ be the least upper bound of $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ and $M_t = (\mathcal{P}_t, \gamma_t)$ be a model such that $\mathcal{P}_t \preceq \mathcal{P}_\top$. Then a translation T such that $\sigma_T(\mathcal{P}_\top) \preceq \mathcal{P}_t$ is a correct translation from any model $M_i \in \mathcal{M}$ to M_t.*

An important consequence of the above result is that in the lattice framework, given a set of models $\mathcal{M}$, there is no need to specify a translation for each pair of models in $\mathcal{M}$ as it is sufficient to look for translations from $\mathcal{P}_\top$ to every model. Thus, the number of required translations is linear in terms of the number of involved models, rather than quadratic. It could be said that the set of patterns $\mathcal{P}_\top$ represents a "supermodel" that contains all the possible constructs used in the various models.

4.2 Completeness of sets of translation rules

Definition 4.1 *A set of basic translation rules $\mathcal{R}$ is* complete *with respect to a set of models $\mathcal{M}$ if for every pair of models M_s and M_t in $\mathcal{M}$ there is a correct translation T based on $\mathcal{R}$ from M_s to M_t.*

By the results on correctness, we can verify this property as follows. We say that a model $M_m = (\mathcal{P}_m, \gamma) \in \mathcal{M}$ is *minimal* in $\mathcal{M}$ if there is no model $M = (\mathcal{P}, \gamma) \in \mathcal{M}$, different from M_m, such that $\mathcal{P} \preceq \mathcal{P}_m$.

Theorem 4.2 *A set of translation rules $\mathcal{R}$ is* complete *with respect to a set of models $\mathcal{M}$ if and only if for each minimal model $M_m = (\mathcal{P}_m, \gamma) \in \mathcal{M}$ there is a translation T such that $\sigma_T(\mathcal{P}_\top) \preceq \mathcal{P}_m$.*

The importance of this result relies also on the fact that it suggests a methodology to achieve completeness. If a set of translation rules turns out to be incomplete because we are not able to find a translation from $\mathcal{P}_\top$ (the lub of our models) to a certain minimal model, then we can add new rules to guarantee that this translation can be done.

4.3 Comparisons between translations

The various characterizations on correctness of translations give rise to a natural measure of the quality of a translation from one model to another. We have observed in [3] that the structures we should obtain as a result of a "good" translation have to exploit as much as possible the constructs of the target model. This notion can be formalized as follows.

Definition 4.2 *Given two different correct translations T_1 and T_2 from a source model $M_s = (\mathcal{P}_s, \gamma_s)$ to a generic target model, T_1* is preferable *than T_2 if $\sigma_{T_2}(\mathcal{P}_s) \preceq \sigma_{T_1}(\mathcal{P}_s)$.*

Thus, a translation is preferable than another if the effect of its execution is "closer" to the target model. For instance, a translation towards a version of the E-R model with n-ary relationships that is able to generate both binary and ternary relationships is preferable than another translation that generates only binary relationships.

Again, preferability of translations can be characterized using the notion of subsumption.

Lemma 4.2 *Let T_1 and T_2 be two different correct translations from a source model $M_s = (\mathcal{P}_s, \gamma_s)$ to a target model $M_t = (\mathcal{P}_t, \gamma_t)$. Then, T_1 is preferable to T_2 if and only if $\sigma_{T_2}(\mathcal{P}_s) \preceq \sigma_{T_1}(\mathcal{P}_s)$.*

We introduce now two reasonable types of "optimization" criteria for translations. The former is related to syntactic aspects (that is, length of the translation), whereas the latter is more involved and takes into account the above notion of preferability. Let us denote with $|T|$ the number of basic translations rules occurring in T.

Definition 4.3 *A correct translation T from a source model M_s to a target model M_t is* minimal *if there is no rule $R \in T$ such that $T - \{R\}$ is preferable than T.*

Definition 4.4 *A correct translation T from a source model M_s to a target model M_t is* optimal *if there is no other correct translation T' from M_s to M_t, such that T' is preferable than T.*

5 Automatic generation of translations

In this section we present a number of results that can be used for deriving correct and (possibly) optimal translations between models.

5.1 Reductions and monotonic translations

In searching for translations between sets of patterns (and so, between models), there is an important point to take into account: in the lattice framework we have defined, it is sufficient to search for translation between sets of patterns $\mathcal{P}_t$ and $\mathcal{P}_s$ such that $\mathcal{P}_t \preceq \mathcal{P}_s$. In fact, by Theorem 4.1, it follows that for any pair of models (or, more generally, for any set of models) we can look for a translation from their least upper bound that, by definition, subsumes both of them. We will call a translation with this property a *reduction*. More specifically, we say that a translation T is a reduction with respect to a model $M = (\mathcal{P}, \gamma)$ if $\sigma_T(\mathcal{P}) \preceq \mathcal{P}$. Note that, a reduction may actually contain steps introducing new patterns, but at the end, it always generates a set of patterns that is subsumed by the original set.

A very general method for generating a reduction from $\mathcal{P}_s$ to $\mathcal{P}_t$ consists in selecting rules that eliminate patterns of $\mathcal{P}_t$ which are not allowed in $\mathcal{P}_s$. Unfortunately, this cannot be done naively since the order in which the rule are selected is crucial. In fact, it may happen that a rule that eliminates a certain pattern P is selected before a rule that eliminates another pattern but, as a side effect, introduces P again. This can be avoided by looking for reductions that enjoy the property of *monotonicity*. Intuitively, a monotonic reduction is one in which if it is never the case that a pattern is eliminated in one step and introduced again in a subsequent step.

More specifically, let $T = R_1, \ldots, R_k$ be a reduction with respect to a model $M = (\mathcal{P}, \gamma)$ and let $T|_j$ denote the reduction $R_1, \ldots, R_j$ composed by the first j rules of T ($j \leq k$). Then, we say that T is *monotonic* with respect to M if, for every $1 \leq i < j \leq k$, it is the case that:

$$(\mathcal{P} - \sigma_{T|_i}(\mathcal{P})) \sqcap \sigma_{T|_j}(\mathcal{P}) \neq \emptyset.$$

We now show that this property can be verified locally, by analyzing the set of rule at disposal. Given a set of rules $\mathcal{R}$, the analysis needs the construction of a graph $G_{\mathcal{R}}$, called the *precedence graph* of $\mathcal{R}$, whose nodes represent the rules in $\mathcal{R}$ and such that there is an edge from a rule $R_i = \sigma_i[\tau_i]$ to a rule $R_j = \sigma_j[\tau_j]$ if $\sigma_i = (P_1^i, P_2^i)$, $\sigma_j = (P_1^j, P_2^j)$ and $P_2^i \sqcap P_1^j \neq \emptyset$. Intuitively, the presence of an edge from R_i to R_j means that R_i must be executed before R_j. For reductions, the following result holds.

Lemma 5.1 *Let T be a reduction with respect to a model M involving the rules $\mathcal{R}$. Then, if for each pair of rules $\sigma_i[\tau_i]$, $\sigma_j[\tau_j]$ in T such that $i < j$, $\sigma_i[\tau_i]$ is not reachable from $\sigma_j[\tau_j]$ in $G_{\mathcal{R}}$, then T is monotonic with respect to M.*

Hence, after the selection of a set of rules that implements our reduction, we need to serialize them according to the above property. We say that a translation that satisfies the condition of Lemma 5.1 is *serial* and that a set of rules $\mathcal{R}$ is *serializable* if it is possible to find a serial translation that involves all the rules in $\mathcal{R}$. The following result easily follows.

Lemma 5.2 *A set of translation rules $\mathcal{R}$ is serializable if and only if $G_{\mathcal{R}}$ is acyclic.*

We present now an interesting result for serializable rules $\mathcal{R}$ which turns out to be very useful in the following: it states that the effect of a serial reduction based on $\mathcal{R}$ (1) is easy to compute and (2) is essentially independent of the serialization chosen. Given a set of rules $\mathcal{R}$ and a set of patterns $\mathcal{P}$ let $Del(\mathcal{R}, \mathcal{P}) = \sqcup_{\forall \sigma[\tau] \in \mathcal{R}}(\mathcal{P} - \sigma(\mathcal{P}))$.

Theorem 5.1 *Let $\mathcal{P}$ be a set of patterns and $\mathcal{R}$ be a set of serializable rules. Then, $\sigma_T(\mathcal{P}) = \mathcal{P} - Del(\mathcal{R}, \mathcal{P})$ for any serial reduction based on $\mathcal{R}$.*

5.2 A method for generating translations

On the basis of the above results, we can derive a methodology for the automatic generation of correct translations between models. The methodology is composed in the following steps:

1. Consider a source set of patterns $\mathcal{P}_s$ and a target set of patterns $\mathcal{P}_t$: as explained above, we can assume, without loss of generality, that $\mathcal{P}_t \preceq \mathcal{P}_s$. Then, in order to derive a reduction from $\mathcal{P}_s$ to $\mathcal{P}_t$, we first select rules whose effect deletes patterns that belong to $\mathcal{P}_s$ but do not belong to $\mathcal{P}_t$. If the effect of the obtained set of rules does not allow to generate the target model, we stop here.

2. In the second part, the set of rules $\mathcal{R}_1$ generated in the first part is (possibly) reduced by deleting the rule $R \in \mathcal{R}_1$ that are "redundant", that is, whose elimination does not affect the correctness of the translation.

3. Then, in the third part we verify whether the set of rules $\mathcal{R}_2$ obtained in the second step is serializable. If they are not, we can try to derive another reduction over the set of rules $\mathcal{R}_b - \{R\}$, where R is a rule in $\mathcal{R}_2$ that belongs to a cycle in $G_{\mathcal{R}_2}$ and so, by Lemma 5.2, causes the rule set to be non-serializable. If the remaining rule set does not yield a correct translation, we stop here.

4. Finally, we can try to derive an optimal rule set. This can be done by searching for a serializable reduction over the set of rules $\mathcal{R}_b - \{R\}$, where R is a rule that belongs to the set obtained in the previous step and deletes patterns that are in $\mathcal{P}_t$. The rationale under this choice is that there could be "finer" functions which are able to replace the work done by R and that do not require the deletion of patterns in the target set of patterns.

In general, according to the results presented in Section 4, we can find correct translations between a generic set of models $\mathcal{M} = \{M_1, \ldots, M_k\}$ such that $M_i = (\mathcal{P}_i, \gamma_i)$, for $i = 1, \ldots, k$, by first finding the lub $\mathcal{P}_\top$ of $\mathcal{P}_1, \ldots, \mathcal{P}_k$, and then applying the methodology above to $\mathcal{P}_\top$ and, in turn, $\mathcal{P}_1, \ldots, \mathcal{P}_k$.

On the basis of this methodology, we have defined in [6] practical algorithms for deriving translations between models and between schemes of different models, in the context of the development of a practical tool based on the framework defined in this paper.

6 Conclusions

In this paper we have presented a formal approach to the problem of translating schemes of different data models. We have introduced the notion of pattern as a graph-theoretic tool for the description of models and we have defined a partial order relationship among sets of patterns that induces a lattice on them. This lattice structure allows us to compare different data models and to define and characterize various interesting properties of translations. On the basis of these results, we have then developed a general methodology for deriving correct translations between models that enjoy the above properties.

We believe that this approach brings a contribution to several problems related to cooperative activities within heterogeneous database frameworks, and is promising for further investigations. From a theoretical point of view, we are currently working on extending the results of this paper to more general cases and testing the various features of the approach in an involved case. From a practical point of view we are working on the development of a first prototype of the metamodel.

References

[1] S. Abiteboul and R. Hull. Restructuring hierarchical database objects. *Theoretical Computer Science*, 62(3):3–38, 1988.

[2] P. Atzeni, G. Ausiello, C. Batini, and M. Moscarini. Inclusion and equivalence between relational database schemata. *Theoretical Computer Science*, 19(2):267–285, 1982.

[3] P. Atzeni and R. Torlone. A metamodel approach for the management of multiple models and the translation of schemes. *Information Systems*, 18(6):349–362, 1993.

[4] P. Atzeni and R. Torlone. Management and translation of heterogeneous database schemes. *Second International East-West Database Workshop, Klagenfurt, 1994*, Wokshops in Computing, Springer–Verlag, pag. 346–355, 1994.

[5] P. Atzeni, R. Torlone. Schema Translation between Heterogeneous Data Models in a Lattice Framework. In *Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6), Atalanta, Georgia*, Chapman & Hall, 1995.

[6] P. Atzeni, R. Torlone. Management of Multiple Models in an Extensible Database Design Tool. In *Fifth International Conference on Extending Database Technology (EDBT '96), Avignon, Lecture Notes in Computer Science 1057*, Springer–Verlag, pages 79–95, 1996.

[7] T. Barsalou and D. Gangopadhyay. M(DM): An open framework for interoperation of multimodel multidatabase systems. In *International Conference on Data Engineering*, pages 218–227, 1992.

[8] C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California, 1992.

[9] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database scheme integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[10] H. Biller. On the equivalence of database schemas – A semantic approach to data translation. *Information Systems*, 4:35–47, 1979.

[11] S.A. Borkin. Data model equivalence. In *Fourth International Conf. on Very Large Data Bases, Berlin*, pages 526–534, 1978.

[12] P.P. Chen. The entity-relationship model: toward a unified view of data. *ACM Trans. on Database Syst.*, 1(1):9–36, March 1976.

[13] M. Gyssens, J. Paredaens, and D. Van Gucht. A graph-oriented object database model. In *Ninth ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 417–424, 1990.

[14] R.B. Hull. Relative information capacity of simple relational schemata. *SIAM Journal on Computing*, 15(3):856–886, 1986.

[15] R.B. Hull and R. King. Semantic database modelling: survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[16] L.A. Kalinichenko. Methods and tools for equivalent data model mapping construction. In *International Conf. on Extending Database Technology (EDBT'90), Lecture Notes in Computer Science 416*, pages 92–119, Springer-Verlag, 1990.

[17] G.M. Kuper. *The Logical Data Model: A New Approach to Database Logic.* PhD thesis, Stanford University, 1985.

[18] Y.E. Lien. On the equivalence of database models. *Journal of the ACM*, 29(2):333–362, 1982.

[19] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Eighteenth International Conf. on Very Large Data Bases*, pages 120–133, 1993.

[20] J. Rissanen. On equivalence of database schemes. In *ACM SIGACT SIGMOD Symp. on Principles of Database Systems*, pages 23–26, 1982.

[21] A.P. Sheth and J.A. Larson. Federated database systems for managing distribuited database systems for production and use. *ACM Computing Surveys*, 22(3):183–236, 1990.

[22] D. Tsichritzis and F.H. Lochovski. *Data Models*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.