Schema Transformations as an Aid in View Integration

Paul Johannesson

Department of Computer and Systems Sciences, Stockholm University
Electrum 230, S-164 40 Kista, Sweden
email: pajo@sisu.se

Abstract. Two important problems in view integration are identifying and merging semantically equivalent but structurally different modelling constructs. One way to address this problem is to standardize the schemas to be integrated by applying a number of schema transformations to them. We formalize the notion of schema transformation in the context of a logic based modelling approach. We then introduce several transformations and show how they can be used in the view integration process.

1 Introduction

Database management systems have been available for more than two decades, mainly in the form of the hierarchical, network, and relational models. In the mid 1970s the development of semantic database models was initiated. These were introduced primarily as schema design tools, meaning that a schema should first be designed in a high level semantic model and then translated into one of the traditional models for implementation. One advantage of using semantic data models in this context is that it simplifies the integration of different user perspectives. In fact, one of the basic reasons for using a database approach instead of a file approach is that it makes it possible to define a coherent view of the data of an organization, which may then be used for serving a number of different user perspectives.

Consequently, an important part of conceptual design is to integrate various conceptual schemas. We will refer to this activity by the term schema integration (or view integration), which is more precisely defined as "the activity of integrating the schemas of existing or proposed databases into a global, unified schema" [Batini86]. Research in the area of schema integration has been carried out since the beginning of the 1980s. A comprehensive survey of the area can be found in [Batini86]. Most of the work has been performed in the context of the relational model, [Biskup86], the functional model [Motro87], or (some extended version of) the ER model, [Larson89], [Spaccapietra92]. The

prevalent approach in schema integration has been to derive, more or less automatically, an integrated schema from a set of integration assertions relating equivalent constructs in the views. The integration assertions typically describe set relationships (equality, inclusion etc.) between the extensions of related entity types or attributes, [Effel84], [Spaccapietra92], [Johannesson91].

A limitation of most schema integration approaches found in the literature is that they provide only very simple types of integration assertions, which are incapable of expressing relationships between structurally different modelling constructs. As an example of two semantically equivalent but structurally different constructs, consider fig. 1.1 and fig. 1.2, where the gender of persons in the first schema is represented using an attribute and in the second schema by means of two subtypes.

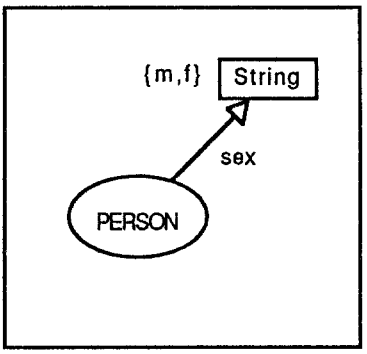

Fig. 1.1

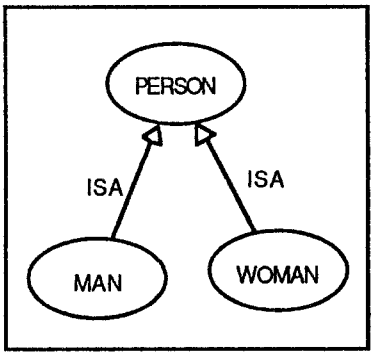

Fig. 1.2

Another problem is that it is usually more difficult to identify relationships involving structurally different constructs than relationships between just two entity types or two attributes. One way to alleviate these problems is to transform the schemas to be integrated before making any integration assertions. The schemas should be transformed so that any relationship between them is expressible as a simple relationship between two entity types or two attributes. So, the goal is to try to standardize the schemas by transforming them into some "normal form" before integrating them. This approach can be compared to the standard schema integration process described in [Batini91], where it is assumed that the schemas shall be modified after they have been compared. The method suggested in this paper reverses comparison and modification by requiring that the schemas first be modified in order to facilitate the schema comparison. The main contribution of this paper is to outline how such an approach can be realized. To this end, we first formalize fundamental concepts pertaining to schema transformations, thereby taking into account static as well as dynamic aspects of a schema. We then present a number of basic schema transformations utilizing the framework proposed. In the next section, we briefly review related work on schema transformations. In section 3, we describe the modelling formalism to be used, which is based on concepts from

logic programming and deductive databases. In section 4, we introduce a number of schema transformations, and in section 5 we show, by means of an example, how they can be applied. In the final section, we summarize the results of the paper and suggest possibilities for further research.

2 Related Research

Most work on schema transformations has been carried out in the context of the relational model. All the first normal form transformations, the third, Boyce-Codd, the fourth, and projection normal form decompositions are examples of schema transformations, [Fagin79], [Ullman88]. A central concern in this work has been to prove that the schema transformations are lossless (information preserving), and several notions of equivalence between relational schemas have been proposed, [Kobayashi86], [Hull86]. Informally, most of these proposals define two schemas as being equivalent if there exists a (simple) bijection between the instances of the schemas. Another important issue has been to show how to transform different types of rules and constraints associated with a relational schema, such as functional dependencies and inclusion dependencies, [Kobayashi86]. This work has been extended by research on transformations in the context of semantic data models, such as the ER model [Chen76], and extensions of the ER model [ElMasri85]. In this research, [Halpin90], [Hainaut91], also more general types of constraints have been taken into consideration, e.g. cardinality constraints and exclusion constraints.

An important difference between the approach taken in this paper and previous work is that we use a logic based formalism of conceptual modelling as a framework. This makes it possible to address more complex design issues arising from the richer semantics of conceptual modelling approaches in contrast to the relational model. Further, we also pay attention to the behaviour of a UoD (Universe of Discourse). The schema transformations presented in the paper are certainly not exhaustive, but we believe they are among the most important arising in practical modelling situations. They also provide an illustration of design problems not occurring in the context of the relational model by focussing on issues such as generalization, object identity, and dynamics.

3 Conceptual Schemas

In this section, we give a formalization of some of the basic concepts in conceptual modelling. The formalization is based on concepts from the logic programming and deductive database areas, [Gallaire84], and it attempts to capture both static and dynamic modelling constructs. We first recall some basic definitions concerning first order languages.

Let P, F, and C be three sets of symbols. A *(first order) language based on* $<P,F,C>$, written $L(P,F,C)$ is defined on an alphabet consisting of connectives, quantifiers, punctuation symbols, variables, function symbols F, constants C, and predicate symbols P, where each predicate symbol has an arity. A (first order)

formula in a language L is defined as usual. A *term* is a constant or a variable. An *atom* is a formula of the form $p(t_1,...,t_n)$, where p is a predicate symbol and $t_1,...,t_n$ are terms. A *literal* is an atom or the negation of an atom. A *ground formula* is a formula without variables. For any language L(P,F,C) we assume that P contains a special symbol, "=", which is interpreted as the identity. A *clause* is a formula of the form $\forall x_1,...,\forall x_n(L_1 \vee ... \vee L_m)$, where each L_i is a literal and $x_1,...,x_n$ are all the variables occurring in $L_1 \vee ... \vee L_m$. We will use the standard notation $A_1,...,A_k \leftarrow B_1,...,B_n$ to denote the clause $\forall x_1,...,\forall x_m(A_1 \vee ... \vee A_k \vee \sim B_1 \vee ... \vee \sim B_n)$. $A_1,...,A_k$ is the *head* of the clause and $B_1,...,B_n$ its *body*. A *definite clause* is a clause of the form $A \leftarrow B_1,...,B_n$, which contains precisely one atom in its consequent and each B_i is an atom.

Since we intend to consider temporal aspects of a UoD, we assume that all languages contain a set of special symbols used to denote points in time. We also assume that there is a total order, denoted by "<", defined over the points in time.

The basic building blocks in all conceptual modelling approaches are objects (entities) and attributes. Objects are often grouped together into object types, such as Employee and Department. To represent that a given object belongs to a certain object type, we use a binary predicate, where the first argument is a constant denoting the object, and the second argument denotes a point in time. As an example, consider *employee(John,T7)*, which expresses that John is an employee at the point in time T7. To represent attribute values, we use ternary predicates. An example is *owns(John,ABC123,T9)*, which expresses that John owns the car ABC123 at the point in time T9.

In general, an *integrity constraint* is defined as any closed first order formula. In this paper, however, we only consider certain special cases of constraints, which frequently occur in conceptual modelling. A *typing constraint* is a formula of the form $\forall x \forall y \forall t(A(x,y,t) \rightarrow D(x,t))$ or the form $\forall x \forall y \forall t(A(x,y,t) \rightarrow R(y,t))$. A formula of the first form will be abbreviated "domain(A) = D", and a formula of the second form as "range(A) = R". We also say that the domain of A is D and the range of A is R. *Mapping constraints* concern the cardinality of attributes and have one of the following four forms. The expression "the attribute A is single-valued" is an abbreviation of the formula $\forall x \forall y \forall z \forall t(A(x,y,t) \wedge A(x,z,t) \rightarrow y = z)$. The expression "the attribute A is injective" is an abbreviation of the formula $\forall x \forall y \forall z \forall t(A(y,x,t) \wedge A(z,x,t) \rightarrow y = z)$. An attribute which is both single-valued and injective is said to be 1-1. The expression "the attribute A is total" is an abbreviation of the formula $\forall x \forall t(P(x,t) \rightarrow \exists y A(x,y,t))$, where P is the domain of A. An attribute which is not total is called partial. The expression "the attribute A is surjective" is an abbreviation of the formula $\forall x \forall t(P(x,t) \rightarrow \exists y A(y,x,t))$, where P is the range of A. A *generalization constraint* is a formula of the form $\forall x \forall t(P(x,t) \rightarrow Q(x,t))$ and is abbreviated $P \subset Q$. We also say that P is a subtype of Q. The expression "the types P1 and P2 are disjoint" is an abbreviation of the formula $\neg \exists x \exists t(P1(x,t) \wedge P2(x,t))$. The expression "the type P has a finite extension" is an abbreviation of the formula

$\forall x \forall t(P(x,t) \rightarrow x = a_1 \vee \ldots \vee x = a_n)$, where $a_1,\ldots,a_n$ are constants. Suppose that $P_1,\ldots,P_n$ are common subtypes to a type Q, then the expression "$P_1,\ldots,P_n$ are exhaustive w.r.t. Q" is an abbreviation of the formula $\forall x \forall t(Q(x,t) \rightarrow P_1(x,t) \vee \ldots \vee P_n(x,t))$.

The constraints above take only static aspects of a UoD into consideration, i.e. they describe what should hold true in each snapshot of the UoD. We now turn to dynamic integrity constraints, which describe how a UoD can evolve over time. The expression "A is domain-stable" is an abbreviation of the formula $\forall x \forall y \forall s \forall t(D(x,s) \wedge D(x,t) \wedge A(x,y,s) \wedge R(y,t) \wedge s<t \rightarrow A(x,y,t))$, where D is the domain of A. Intuitively, an attribute is domain stable if an object which acquires a value for the attribute keeps that value for its complete life time. The dual of domain-stability is range-stability: The expression "A is range-stable" is an abbreviation of the formula $\forall x \forall y \forall s \forall t(R(x,s) \wedge R(x,t) \wedge A(y,x,s) \wedge D(y,t) \wedge s<t \rightarrow A(y,x,t))$, where R is the range of A. Suppose that P is a subtype of Q, then the expression "P is a stable subtype of Q" is an abbreviation of the formula $\forall x \forall t(P(x,t) \rightarrow \forall s(Q(x,s) \rightarrow P(x,s)))$. For an example of the concept of stable subtype, see section 4.5.

A conceptual schema is usually informally defined as an implementation independent description of the contents in an information system. We here define a *conceptual schema* as a pair <L,IC>, where L is a language and IC is a set of integrity constraints of the form given above. All predicate symbols in L are assumed to be either binary or ternary, and for each ternary predicate symbol p, the existence of two constraints in IC specifying the domain and range of p is assumed. In the following, we shall call the binary predicate symbols "object types" and the ternary "attributes". We assume there exist a set of predicate symbols LP and a set of constant symbols LC (called *lexical predicate and constant symbols)*, such that for any conceptual schema its language L is based on <P∪ LP,LC>, where P is a set of predicate symbols.

The motive for introducing the lexical predicate symbols and constants is to capture the distinction between object identifiers (surrogates) and data values [Beeri89]. The point of this distinction is that in all interpretations, the data values are to denote the same objects, whereas object identifiers may denote different objects in different interpretations. As an example, the value "7" should always denote the natural number seven, while an object identifier "qz27" may denote an employee in one interpretation and a department in another. Usually, the values are integers, strings, booleans etc.

In fig. 3.1, a graphical representation of a conceptual schema is shown. Note that the graph only depicts a part of the language of the schema and some integrity constraints. The graph shows that the schema contains five binary predicate symbols {person, man, woman, pet, string} and five ternary {ss#, name, sibling, possesses, married_to}, corresponding to object types and attributes, respectively. The only lexical predicate symbol occurring in this schema is "String", shown in the graph as a rectangle. The graph also shows a number of typing constraints, e.g. "domain(possesses) = PERSON" and "range(possesses) = PET". The generalization constraints MAN ⊂ PERSON and WOMAN ⊂

PERSON are shown by arcs labelled ISA. There are also some additional integrity constraints, which are not shown graphically: *married_to* is single-valued and injective; *possesses* is injective and surjective; *sibling* is domain-stable and range-stable; *ss#* is total, single-valued, and injective; *name* is total and single-valued.

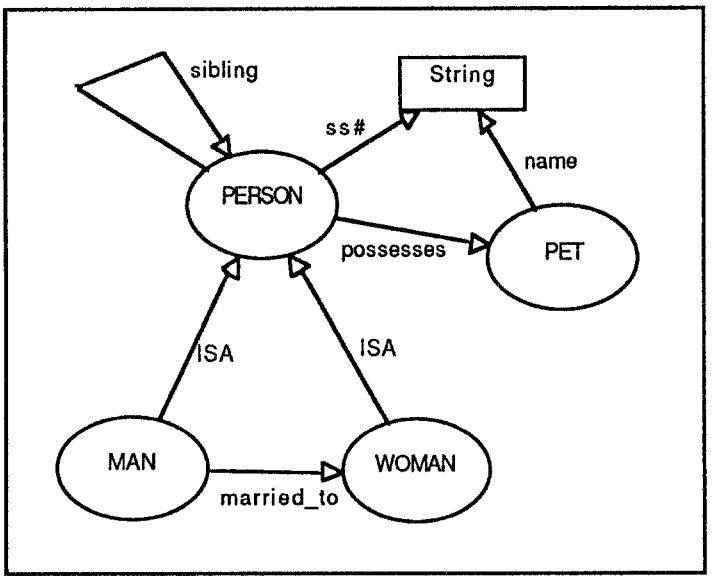

Fig. 3.1 Example of a conceptual schema

We now turn to the definition of an information base; informally an information base contains information about particular objects and associations between these. An *information base* for a conceptual schema CS = <L(P,LC), IC> is a pair <C, F>, where C is a finite set of constants, and F is a finite set of ground atoms whose predicate symbols belong to P and whose constants belong to LC U C. Note that an information base can be viewed as an interpretation of L over the Herbrand universe given by LC U C. We write IB_{CS} to denote the set of all information bases for CS.

The role of integrity constraints is to state conditions that must hold true for each information base: Let CS = <L,IC> be a conceptual schema and IB = <C, F> an information base for CS. The information base IB *violates* the schema CS if some constraint in IC is not true in F.

An example of a small information base for the schema in fig. 3.1 is the following. Note that the information base does not violate the schema supposing t1 < t2.

{man(p1,t1), person(p1,t1), ss#(p1,'4511010098',t1),
string('4511010098',t1), man(p1,t2), person(p1,t2),
ss#(p1,'4511010098',t2), string('4511010098',t2), pet(e1,t2),
possesses(p1,e1,t2)}

This representation is obviously not adequate for implementing real databases, but it does provide a convenient formal framework for analyzing schema transformation issues.

4 Basic Schema Transformations

In this section, we formalize the concept of schema transformation and present a number of basic transformations. Let C be the set of all conceptual schemas. A *schema transformation* is a function from C to C. We introduce a requirement that a transformed schema shall fulfil w.r.t. the original schema. Informally, the requirement says that the transformed schema shall be able to represent at least as much information as the original one. We first have to introduce some notation. If P is a set of definite clauses, then M_P denotes the least Herbrand model for P. Let L1(P1,F1,C1) and L2(P2,F2,C2) be two languages such that L1 is an expansion of L2, i.e. $P2 \subset P1$, $F2 \subset F1$, and $C2 \subset C1$. Let H be a Herbrand interpretation of L1. The *restriction of H to L2*, denoted $H|L2$, is the subset of H whose elements only involve predicate symbols from L2. Let M and N be two Herbrand interpretations, and let M_t and N_t be the terms in M and N, respectively. M and N are *isomorphic*, denoted by $M \cong N$ if there is a bijection α: $M_t \rightarrow N_t$, and M' = N, where M' is obtained from M by substituting each term t in M with $\alpha(t)$.

Let CS1 = <L1,IC1> and CS2 = <L2,IC2> be two conceptual schemas. CS1 *dominates* CS2 if there exist a function $\Phi: IB_{CS2} \rightarrow IB_{CS1}$ and a set D of definite clauses, such that the heads of the clauses in D are expressed in L2 and the bodies in L1, and

(i) Φ is total and injective

(ii) if IB2 does not violate CS2, then $\Phi(IB2)$ does not violate CS1

(iii) for every information base IB2 = <C2,F2> of CS2, $M_{\Phi(F2)} \cup D|L2 \cong M_{F2}$.

Informally, a schema CS1 dominates another schema CS2 if there exists a simple total mapping from the information bases of CS1 to the information bases of CS2. By "simple", in this context, we mean that the mapping can be defined by a set of definite clauses. We say that two schemas, CS1 and CS2, are *equivalent* if CS1 dominates CS2 and CS2 dominates CS1.

In the following subsections, we present a number of basic schema transformations. We also prove that a schema produced by any of the transformations dominates the original schema.

4.1 A Transformation for Partial Attributes

Two equivalent schemas may differ in their use of subtypes. One way to standardize the use of subtypes is to require all attributes to be total. If an attribute in a schema is partial, the schema can be transformed by introducing a new subtype so that the attribute becomes total. An example is shown in fig. 4.1, where the type PERSON is the domain of a partial attribute *salary*. To make

the attribute total, we introduce a new subtype, EMPLOYEE, as shown in fig. 4.2.

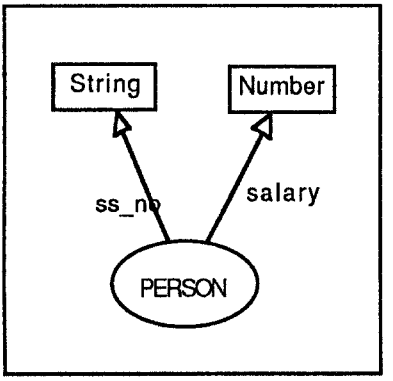

Fig. 4.1

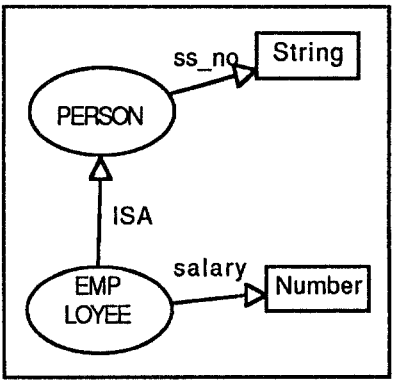

Fig. 4.2

Transformation 1:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains an attribute **p**, which is not total, i.e. "**p** is total" $\notin$ IC. Let **q** be the domain of **p** and let **qs** be a type not belonging to P. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup \{qs\},F,C)$
$IC \mapsto IC \cup \{p \text{ is total, } domain(p) = qs, qs \subset q\} - \{domain(p) = q\}$

4.2 A Transformation for m-m Attributes

When modelling associations between objects, one often has a choice between two alternatives. Either the association can be represented by a single attribute, or an extra object used to connect the associated objects can be introduced. An example where the association is represented by a single attribute is given in fig. 4.3, where the fact that persons can own cars is modelled by an attribute *owns*. An equivalent schema is shown in fig. 4.4, where the association between cars and persons is modelled with the help of an additional type OWNERSHIP. To standardize schemas in this respect, a restriction can be placed on the schemas that no m-m attributes (i.e. attributes which are neither single-valued nor injective) be allowed. If an m-m attribute occurs in a schema, it can be transformed by introducing a new object type.

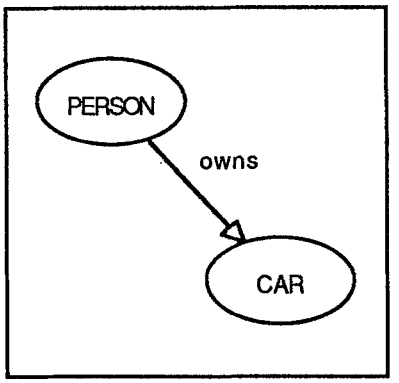

Fig. 4.3

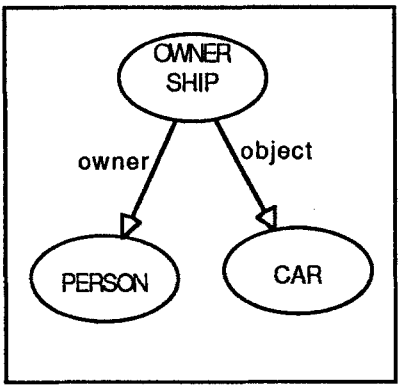

Fig. 4.4

Transformation 2:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when CS contains an m-m attribute p with domain d and range e. Let q be a type and p_1, p_2 two attributes such that $\{q, p_1, p_2\} \cap P = \emptyset$. Let f denote a function symbol not belonging to F. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup \{q,p_1,p_2\} - \{p\}, F \cup \{f\}, C)$
$IC \mapsto IC - \{ic \in IC \mid ic \text{ concerns } p\} \cup$
$\{domain(p_1) = q, range(p_1) = d, domain(p_2) = q, range(p_2) = e\} \cup$
$\{p_1$ is single-valued, total, and domain-stable$\} \cup$
$\{p_2$ is single-valued, total, and domain-stable$\} \cup$
$\{p_1$ is injective $\mid$ p is single-valued$\} \cup$
$\{p_2$ is injective $\mid$ p is injective$\} \cup$
$\{p_1$ is surjective $\mid$ p is total$\} \cup$
$\{p_2$ is surjective $\mid$ p is surjective$\} \cup$
$\{p_1$ is range-stable $\mid$ p is domain-stable$\} \cup$
$\{p_2$ is range-stable $\mid$ p is range-stable$\} \cup$

4.3 A Transformation for Lexical Attributes

A frequent cause for differences between schemas describing the same UoD is that the same phenomenon can often be modelled by either a non-lexical object type or by a lexical one. As an example, consider figures 4.5 and 4.6, which show two equivalent schemas. In fig. 4.5, the fact that cars have colours is modelled as an association between cars and strings, whereas in fig. 4.6, the same fact is modelled as an association between cars and colour objects. To avoid discrepancies of this type between schemas, one can request that lexical objects should only be used as names for other objects, i.e. by requiring that every

attribute with a lexical type as range should be 1-1. If a schema contains an attribute with a lexical type as range and which is not 1-1, then the schema can be transformed by introducing a new object type, which becomes the range of that attribute.

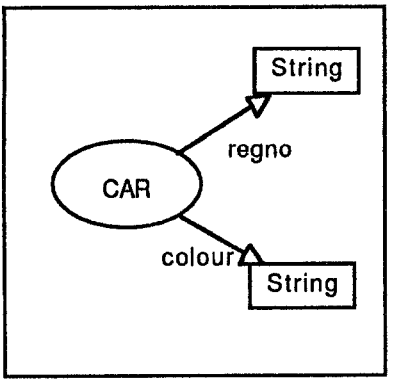

Fig. 4.5

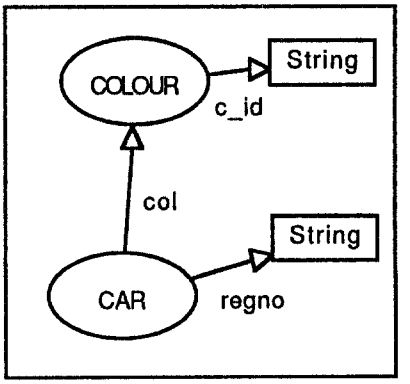

Fig. 4.6

Transformation 3:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains an attribute p, which has a lexical type lt as range and which is not 1-1. Let q be a type and q_id an attribute such that {q, q_id} ∩ P = ∅. Let $f \notin F$ be a function symbol. Let $R = \{r_1,...,r_n\} \subset P$ be the set of all attributes with lt as range. Let $R' = \{r_1',...,r_n'\}$ be a set of attributes such that $R' \cap P = \emptyset$. The transformation is given by:

L(P,F,C) ↦ L(P ∪ {q,q_id} ∪ R' - R,F ∪ {f},C)
IC ↦ IC - {ic ∈ IC | ic concerns an attribute in R} ∪
{domain(q_id) = q, range(q_id) = lt} ∪
{q_id is single-valued, injective, total, and surjective} ∪
{range(r_i') = q | $r_i \in R$} ∪
{r_i' is single-valued (injective, total, surjective, domain-stable, range-stable) |
$r_i \in R$, r_i is single-valued (injective, total, surjective, domain-stable, range-stable)}

4.4 A Transformation for Attributes with Fixed Ranges

It is possible to use attributes for classifying objects into different categories. This can be done when the range of the attribute contains a small, fixed number of values. An example is shown in fig. 4.7, where persons are classified into two different groups, based on sex. An equivalent schema is given in fig. 4.8, where the attribute sex has been replaced by two subtypes to PERSON. To avoid

discrepancies of this type between schemas, one can require that attributes with ranges, that have a small, finite extension not be allowed. If a schema should contain such an attribute, it could be transformed by introducing a number of subtypes to the domain of the attribute. These subtypes should correspond to the values of the range of the attribute.

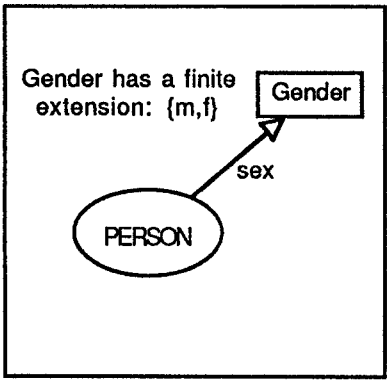

Fig. 4.7

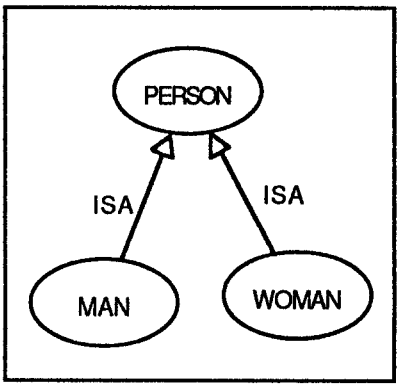

Fig. 4.8

Transformation 4:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains an attribute p with domain d and range lt, where lt is a lexical type with a finite extension $\{a_1,...,a_n\}$. Further, suppose that p is domain-stable, but neither injective, surjective, nor range-stable. Let $Q = \{q_1,...,q_n\}$ be a set of n types such that $P \cap Q = \emptyset$. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup Q - \{p\}, F, C)$
$IC \mapsto IC - \{ic \in IC \mid ic \text{ concerns } p\} \cup$
$\{q_i \subset d \mid q_i \in Q\} \cup$
$\{q_1,...,q_n \text{ are exhaustive w.r.t. } d \mid p \text{ is total}\} \cup$
$\{q_i \text{ and } q_j \text{ are disjoint} \mid q_i, q_j \in Q, i \neq j, p \text{ is single-valued}\} \cup$
$\{q_i \text{ is a stable subtype of } d \mid q_i \in Q\}$

4.5 A Transformation for Stable Subtypes

Another cause for discrepancies between schemas describing the same UoD is that sometimes a certain phenomenon can be modelled as either a subtype of a given object type or as a type associated via an attribute to the given type. As an example, consider figures 4.9 and 4.10. In fig. 4.9, the fact that persons can be employed by companies is modelled by the use of a subtype EMPLOYEE. In fig. 4.10, the same fact is modelled by a type EMPLOYMENT and a one-to-one attribute *of*. It can be noted that being an employee is not an intrinsic property of a person. Instead, being an employee can be regarded as a role played by a

person in relation to a company; a person can start and stop being an employee. Note that a person must remain a person during his complete life-time, whereas he may become and stop being an employee several times during his life-time. In other words, EMPLOYEE is not a stable subtype of PERSON. To standardize schemas, we could require that only stable subtypes should be allowed. If a schema contains a non-stable subtype, we could replace the ISA-relation by a 1-1 attribute as described in the following transformation.

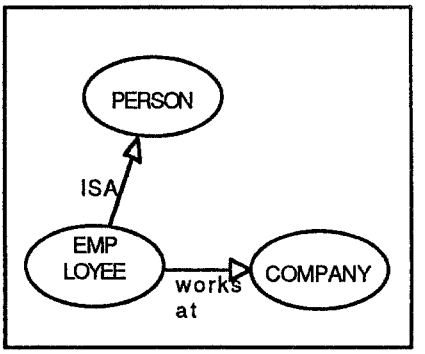

Fig. 4.9

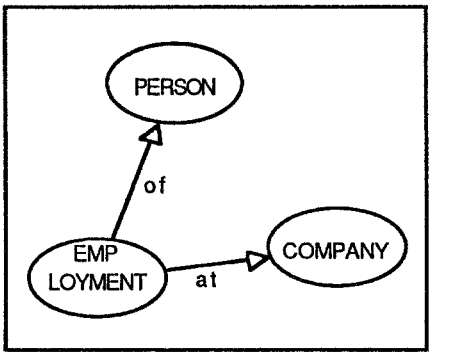

Fig. 4.10

Transformation 5:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains two types q and qs such that $q \subset qs$, but q is not a stable subtype of qs. Let $q' \notin P$ be a type. Let $R = \{r_1,...,r_n\}$ be the set of all attributes with q as domain. Let $R' = \{r_1',...,r_n'\}$ be a set of attributes such that $R' \cap P = \emptyset$. Let $\{s_1,...,s_m\}$ be the set of supertypes of q. Let $T = \{t_1,...,t_m\}$ be a set of attributes such that $T \cap P = \emptyset$. Let $f \notin F$ be a function symbol. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup R' \cup T \cup \{q'\} - R - \{q\}, F \cup \{f\}, C)$
$IC \mapsto IC$ - {ic ∈ IC | ic concerns q or an attribute in R} ∪
{domain(r_i') = q' | $r_i \in R$} ∪
{range(r_i') = u_i | $r_i \in R$, range(r_i) = u_i} ∪
{domain(t_i) = q' | $t_i \in T$} ∪
{range(t_i) = s_i | $t_i \in T$} ∪
{t_i is single-valued, injective, and total | $t_i \in T$} ∪
{r_i' is single-valued (injective, total, surjective, domain-stable, range-stable) |
$r_i \in R$, r_i is single-valued (injective, total, surjective, domain-stable, range-stable)}

4.6 A Transformation for Lattice Structure

If two types are not disjoint, it may be useful to introduce a common subtype, whose extension is the intersection of the extensions of the two types. As an example, consider figures 4.11 and 4.12, where a subtype AMPHIBIOUS_VEHICLE of the types CAR and BOAT is introduced. We could then require that whenever there are two types which are not disjoint, they should have a common subtype.

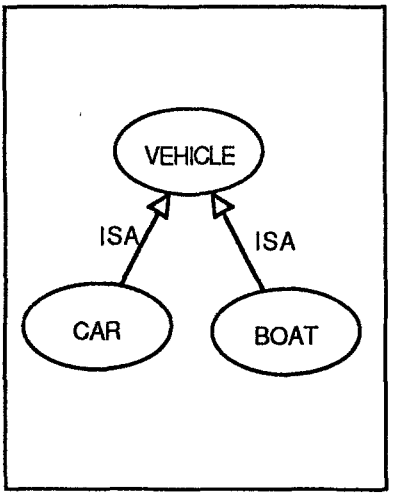

Fig. 4.11

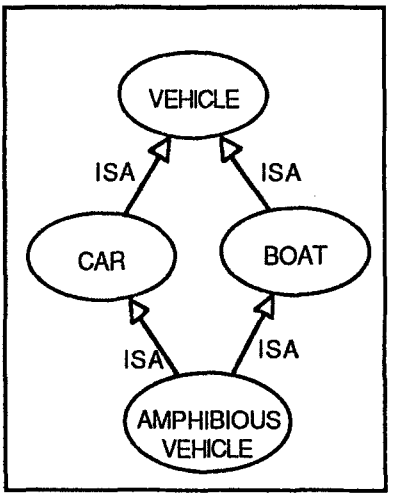

Fig. 4.12

Transformation 6:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains two types p and q, which are not disjoint. Let $r \notin P$ be a type. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup \{r\},F,C)$
$IC \mapsto IC \cup \{r \subset p, r \subset q\}$

Proposition 4.1: For transformations 1 - 4 above, the transformed schema is equivalent to the original schema. For transformations 5 - 6 above, the transformed schema dominates the original schema.

Proof Sketch: Let S be a conceptual schema and S_i the schema obtained by applying transformation i to S. In accordance with the definition of dominance in the beginning of section 4, we give functions Φ_i: $IB_S \rightarrow IB_{Si}$ and Ψ_i: $IB_{Si} \rightarrow IB_S$ and corresponding sets of definite clauses D_i and E_i.

Transformation 1:

$\Phi_1(IB) = IB \cup \{qs(a,t) \mid \exists b\, p(a,b,t) \in IB\}$

$D_1 = \emptyset$

$\Psi_1(IB) = IB - \{qs(a,t) \mid qs(a,t) \in IB\}$

$E_1 = \{qs(a,t) :- p(a,b,t)\}$

Transformation 2:

$\Phi_2(IB) = IB - \{p(a,b,t) \mid p(a,b,t) \in IB\} \cup \{q(f(a,b),t), p_1(f(a,b),a,t), p_2(f(a,b,t),b) \mid p(a,b,t) \in IB\}$

$D_2 = \{p(x,y,t) :- q(f(x,y),t)\}$

$\Psi_2(IB) = IB - \{q(a,t) \mid q(a,t) \in IB\} - \{p_1(a,b,t) \mid p_1(a,b,t) \in IB\} - \{p_2(a,b,t) \mid p_2(a,b,t) \in IB\} \cup \{p(a,b,t) \mid \exists s(p_1(s,a,t) \in IB, p_2(s,b,t) \in IB\}$

$E_2 = \{q(f(x,y), t) :- p(x,y,t), p_1(f(x,y), x, t) :- p(x,y,t), p_2(f(x,y), y, t) :- p(x,y,t)\}$

Transformation 3:

$\Phi_3(IB) = IB - \{r_i(a,b,t) \mid r_i(a,b,t) \in IB, r_i \in R\} \cup \{r_i'(a,f(b),t) \mid r_i(a,b,t) \in IB, r_i \in R\} \cup \{q(f(a),t), q\_id(f(a), a, t) \mid lt(a,t) \in IB\}$

$D_3 = \{r_i(x,y,t) :- r_i'(x,f(y),t) \mid r_i \in R\}$

$\Psi_3(IB) = IB - \{r_i'(a,b,t) \mid r_i'(a,b,t) \in IB, r_i' \in R'\} - \{q(a,b) \mid q(a,b) \in IB\} - \{q\_id(a,b,t) \mid q\_id(a,b,t) \in IB\} \cup \{r_i(a,b,t) \mid \exists s(r_i'(a,s,t) \in IB, q(s,t) \in IB, q\_id(s,b,t) \in IB, r_i' \in R'\}$

$E_3 = \{r_i'(x,f(y),t) :- r_i(x,y,t), q(f(x),t) :- lt(x), q\_id(f(x),x) :- lt(x)\}$

Transformation 4:

$\Phi_4(IB) = IB - \{p(a,b,t)) \mid p(a,b,t)) \in IB\} \cup \{q_i(a,t) \mid p(a,a_i,t) \in IB\}$

$D_4 = \{p(x,a_i,t) :- q_i(x,t) \mid q_i \in Q\}$

$\Psi_4(IB) = IB - \{q_i(a,t) \mid q_i(a,t) \in IB, q_i \in Q\} \cup \{p(a,a_i,t) \mid q_i(a,t) \in IB, q_i \in Q\}$

$E_4 = \{q_i(x,t) :- p(x,a_i,t)\}$

Transformation 5:

$\Phi_5(IB) = IB - \{q(a,t) \mid q(a,t) \in IB\} -$
$\{r_i(a,b,t) \mid r_i(a,b,t) \in IB, r_i \in R\} \cup$
$\{q'(f(a),t) \mid q(a,t) \in IB\} \cup$
$\{t_i(f(a),a,t) \mid q(a,t) \in IB, t_i \in T\} \cup$
$\{r_i'(f(a),b,t) \mid r_i \in R, r_i(a,b,t) \in IB\}$

$D_5 = \{q(x) :- q'(f(x),t)\} \cup \{r_i(x,y,t) :- r_i'(f(x),y,t) \mid r_i \in R\}$

Transformation 6:

$\Phi_6(IB) = IB \cup \{r(a,t) \mid p(a,t) \in IB, q(a.t) \in IB\}$

$D_6 = \emptyset$

In the appendix below, we give the inverses of the schema transformations presented in this section.

5 An Example

In this section, we show, by means of an example, how the transformations introduced above can be applied. Our point of departure is the schema depicted in fig. 5.1, which can be used to represent information about people living in Ancient Greece. People are assumed to live at addresses and to be citizens in countries. The attribute *sex* is used to represent the gender of a person, and the boolean attribute *human* specifies if a person is a human or an immortal. The figures following fig. 5.1 show how the original schema is successively transformed.

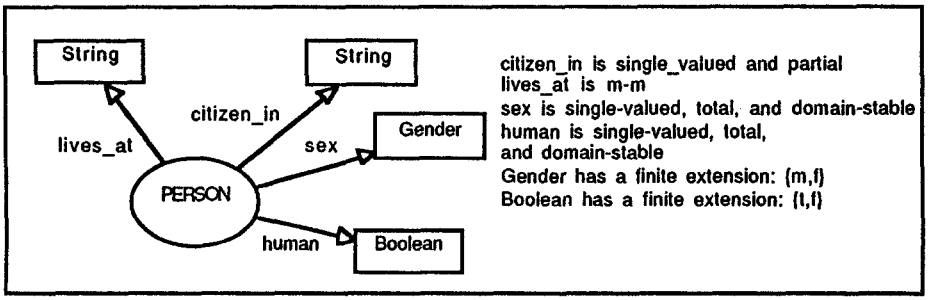

Fig. 5.1 The original schema

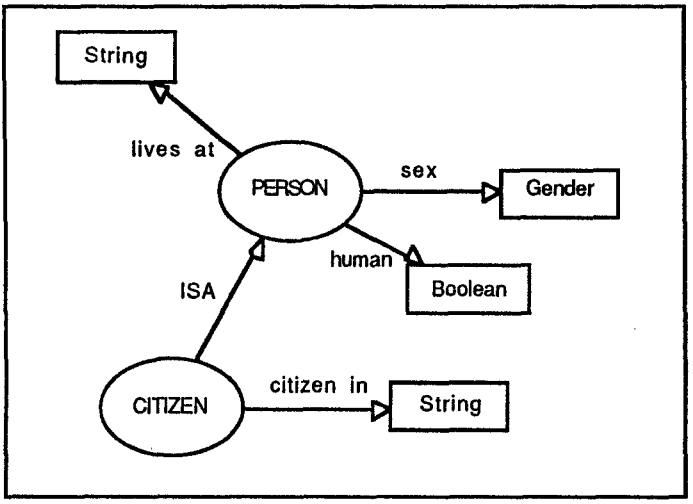

Fig. 5.2 The schema after applying transformation 1 (partial attributes)

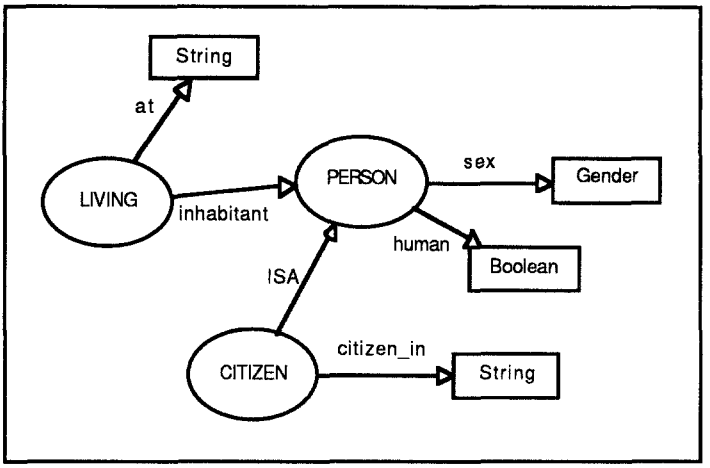

Fig. 5.3 The schema after applying transformation 2 (m-m attributes)

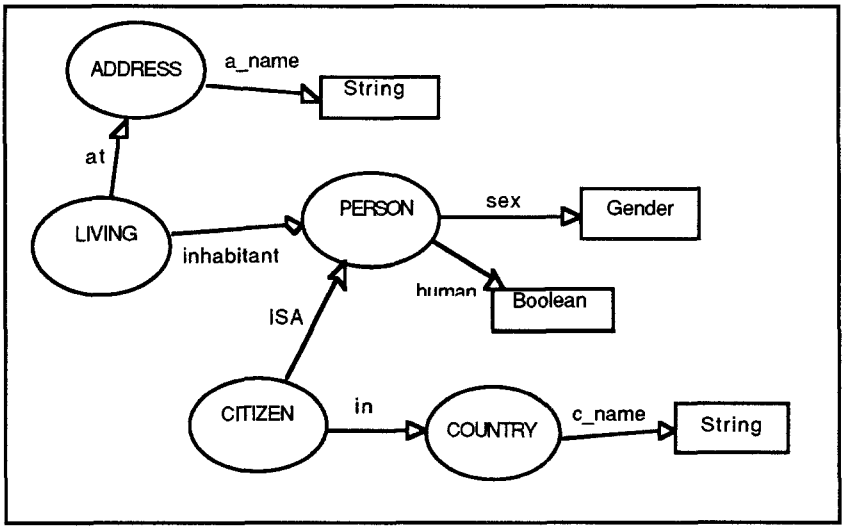

Fig. 5.4 The schema after applying transformation 3 (lexical attributes)

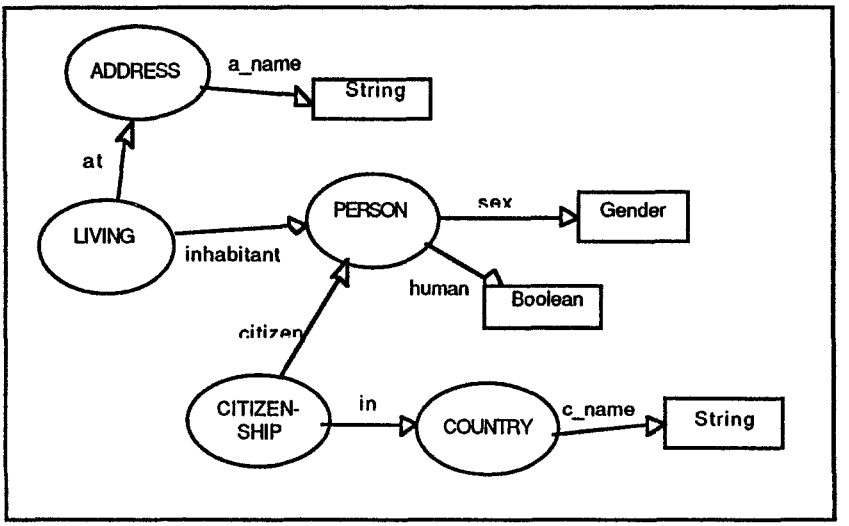

Fig. 5.5 The schema after applying transformation 5 (stable subtypes)

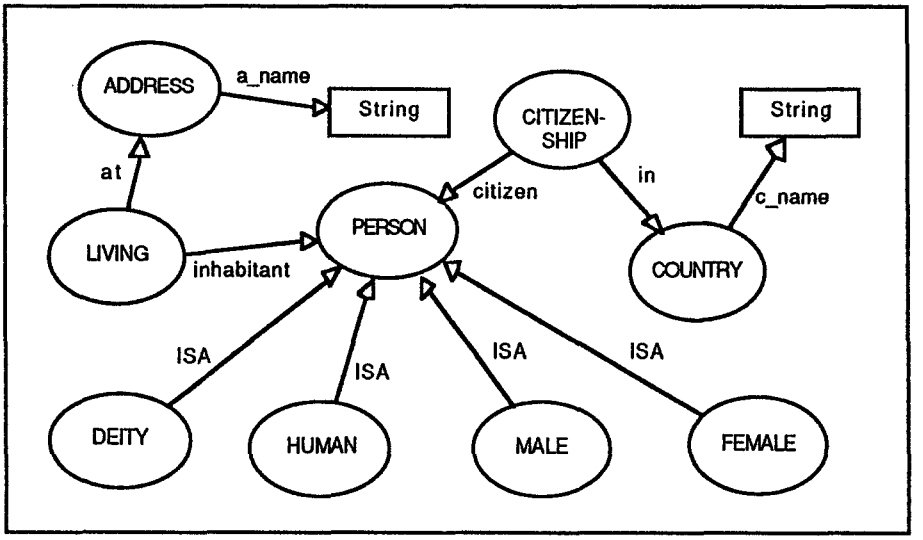

Fig. 5.6 The schema after applying transformation 4 (attributes with fixed ranges)

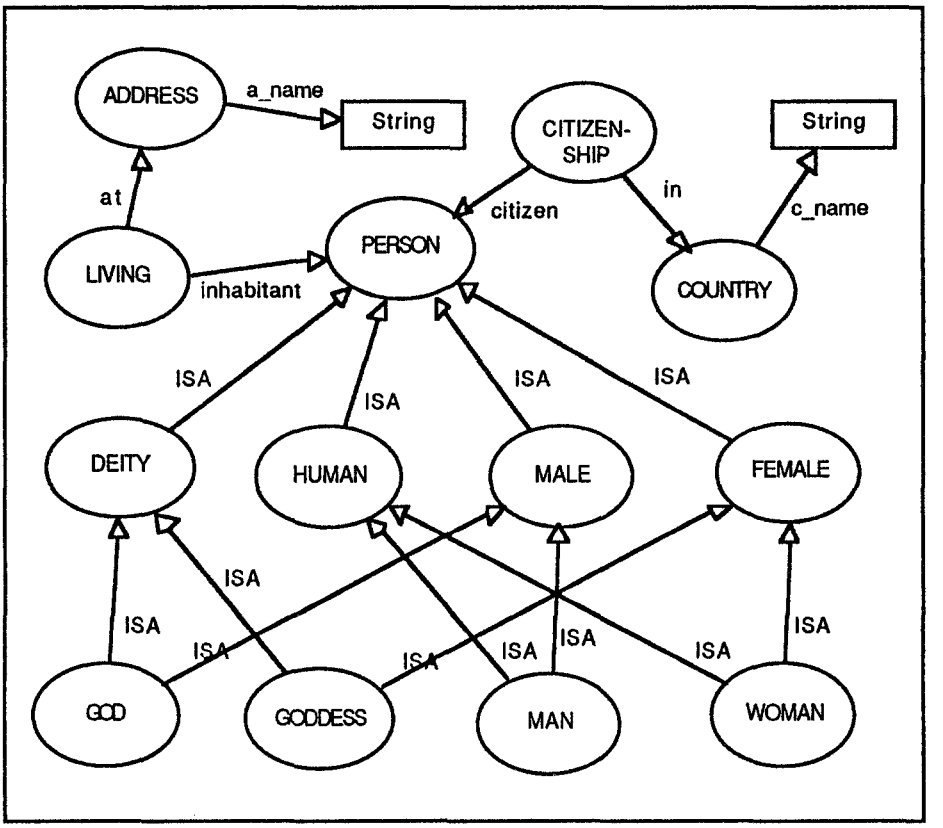

Fig. 5.7 The schema after applying transformation 6 (lattice structure)

As can be seen from the example above, the transformations do not necessarily produce a schema which is more "natural" or intuitively appealing than the original one. However, the schema produced is more standardized in the sense that it reifies as many phenomena of the UoD as possible, i.e. for every phenomenon to be modelled, the schema uses as many object types as possible. For example, the fact that persons can be citizens in countries was modelled in the original schema by means of a single attribute, while in the final schema two object types, CITIZENSHIP and COUNTRY, have been added. Standardizing schemas in this way supports the schema integration process. Since a larger number of the correspondences between the schemas will be simple relationships between two object types or two attributes, it is easier to identify and express relationships between different schemas,

6 Concluding Remarks

In this paper, we have discussed how schema transformations can be used to standardize schemas, and we have indicated how this standardization can

facilitate the view integration process. We have proposed a formal framework for the representation of conceptual schemas based on notions from logic programming and deductive databases. We have also formalized the notion of a schema transformation, and we have presented a number of schema transformations using the framework proposed.

A common property of the schema transformations presented in this paper is that they transform a given schema into a schema satisfying some constraint. For example, the transformation of section 4.1 produces a schema in which a given attribute has to be total. Generating schemas which satisfy certain constraints is typical for most schema transformations found in the literature, e.g. the well-known transformations used for normalization in relational database design produce schemas with certain restrictions on the functional dependencies in the schema. Another characteristic of most of the transformations given in this paper is that they expand the language of the original schema by introducing additional object types and/or attributes. This property makes the transformations useful for applications other than just view integration, e.g. in constructing a natural language interface to a database, the transformations can be used to systematically explore alternative ways of expression and to discover terms not present in the database schema.

The schema transformations proposed in this paper are not meant to be exhaustive, and a direction for future work is to identify additional transformations.. A promising approach here seems to be to investigate transformations that create schemas satisfying dynamic constraints. For example, one could require that destructive updates be forbidden in the transformed schema, which would amount to creating a temporal deductive schema based on an operational schema.

References

[Batini86] C. Batini, M. Lenzerini and S. B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, vol. 18, no. 4, pp. 323-364, 1986.

[Batini91] C. Batini, S. Ceri, and S. Navathe, *Conceptual Database Design*, Benjamin/Cummings, 1991

[Beeri89] C. Beeri, "Formal Models for Object Oriented Databases", in *First International Conference on Deductive and Object Oriented Databases*, Ed. W. Kim, pp. 405-430, Kyoto, North-Holland, 1989.

[Biskup86] J. Biskup and B. Convent, "A Formal View Integration Method", in *International Conference on the Management of Data*, Washington, ACM, 1986.

[Chen76] P. P. Chen, "The Entity-Relationship Model - Toward a Unified View of Data", *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9-36, 1976.

[Effel84] W. Effelsberg and M. V. Mannino, "Attribute Equivalence in Global Schema Design for Heterogenous Distributed Databases", *Information Systems*, vol. 9, no. 3/4, pp. 237-240, 1984.

[ElMasri85] R. ElMasri, J. Weeldryer and A. Hevner, "The Category Concept: An Extension to the Entity-Relationship Model", *Data and Knowledge Engineering*, vol. 1, no. 1, 1985.

[Fagin79] R. Fagin, "Normal Forms and Relational Database Operators", in *ACM SIGMOD*, pp. 153-160, 1979.

[Gallaire84] H. Gallaire, J. Minker and J. M. Nicholas, "Logic and Databases: A Deductive Approach", *ACM Computing Surveys*, vol. 16, no. 2, 1984.

[Hainaut91] J.-L. Hainaut, "Entity Generating Schema Transformations", in *10th International Conference on Entity-Relationship Approach*, San Francisco, 1991.

[Halpin90] T. Halpin, "A Fact-oriented Approach to Schema Transformations", in *Mathematical Foundations of Database Systems*, Springer, 1990.

[Hull86] R. Hull, "Relative Information Capacity of Simple Relational Database Schemata", SIAM Journal of Computing, vol. 15, no. 3, pp. 856-886, 1986.

[Johannesson91] P. Johannesson, "A Logic Based Approach to Schema Integration", in *10th International Conference on Entity-Relationship Approach*, Ed. T. Teorey, San Fransisco, North-Holland, 1991.

[Kobayashi86] I. Kobayashi, "Losslessness and Semantic Correctness of Database Schema Transformations", *Information Systems*, vol. 11, no. 1, 1986.

[Larson89] J. A. Larson, S. Navathe and R. ElMasri, "A Theory of Attribute Equivalence in Databases with Apllications to Schema Integration", *IEEE Transactions on Software Engineering*, vol. 15, no. 4, pp. 449-463, 1989.

[Motro87] A. Motro, "Superviews: Virtual Integration of Multiple Databases", *IEEE Transactions on Software Engineering*, vol. 13, no. 7, pp. 785-798, 1987.

[Spaccapietra92] S. Spaccapietra, C. Parent and Y. Dupont, "Model Independent Assertions for Integration of Heterogeneous Schemas", *The VLDB Journal*, vol. 1, no. 2, pp. 81-126, 1992.

[Ullman88] J. Ullman, *Principles of Database and Knowledge-base Systems*, Computer Press, 1988.

APPENDIX, Inverse Transformations

Transformation 1^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains two types q and qs such that $qs \subset q$ and qs is not the range of any attribute. Let $P = \{p_1,...,p_n\}$ be the set of all attributes whose domains are qs. The transformation is given by:

$L(P,F,C) \mapsto L(P - \{qs\},F,C)$
$IC \mapsto IC \cup \{domain(p_i) = q \mid p_i \in P\} - \{ic \in IC \mid ic \text{ concerns } qs\} - \{p_i \text{ is total} \mid p_i \in P\}$

Transformation 2^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when CS contains a type q that is the domain of exactly two attributes p_1 and p_2 which are single-valued, total, and domain-stable. The range of p_1 is d, and the range of p_2 is e, and both d and e are non-lexical object types. Let p be an attribute such that $p \notin P$. The transformation is given by:

$L(P,F,C) \mapsto L(P \cup \{p\} - \{q,p_1,p_2\}, F, C)$
$IC \mapsto IC - \{ic \in IC \mid ic \text{ concerns } q, p_1, \text{ or } p_2\} \cup$
$\{domain(p) = d, range(p) = e\} \cup$
$\{p \text{ is single-valued} \mid p_1 \text{ is injective}\} \cup$
$\{p \text{ is injective} \mid p_2 \text{ is injective}\} \cup$
$\{p \text{ is total} \mid p_1 \text{ is surjective}\} \cup$
$\{p \text{ is surjective} \mid p_2 \text{ is surjective}\} \cup$
$\{p \text{ is domain-stable} \mid p_1 \text{ is range-stable}\} \cup$
$\{p \text{ is range-stable} \mid p_2 \text{ is range-stable}\}$

Transformation 3^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains a type q that is the domain of exactly one attribute q_id, whose range is a lexical type lt, and q_id is single-valued, injective, total, and surjective. Let $R = \{r_1,...,r_n\} \subset P$ be the set of all attributes with q as range. The transformation is given by:

$L(P,F,C) \mapsto L(P - \{q,q\_id\}, F, C)$
$IC \mapsto IC - \{ic \in IC \mid ic \text{ concerns } q \text{ or } q\_id\} \cup$
$\{range(r_i) = lt \mid r_i \in R\}$

Transformation 4^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains a type d and a set $Q = \{q_1,...,q_n\}$, where each q_i is a stable subtype of d and no q_i is the domain or range of any attribute. Let lt be a lexical type with a finite extension $\{a_1,...,a_n\}$, and let p be a new attribute. The transformation is given by:

L(P,F,C) ↦ L(P ∪ {p, lt} - Q, F, C)
IC ↦ IC - {ic ∈ IC | ic concerns q_i | $q_i \in Q$} ∪
{p is total | $q_1,...,q_n$ are exhaustive w.r.t. d} ∪
{p is single-valued | q_i and q_j are disjoint, $q_i, q_j \in Q$, $i \neq j$} ∪
{p is domain-stable}

Transformation 5^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains a type q that is the domain of at least one attribute that is single-valued, injective, and total. Let $R = \{r_1,...,r_n\}$ be the set of all attributes with q as domain. Let $T = \{t_1,...,t_m\}$ be the subset of R which contains those attributes that are single-valued, injective, and total. Let $R' = \{r_1',...,r_n'\}$ be a set of attributes such that $R' \cap P = \emptyset$, and let q' be an attribute such that $q' \notin P$. The transformation is given by:

L(P,F,C) ↦ L(P - R - {q} ∪ R' ∪ {q'}, F, C)
IC ↦ IC - {ic ∈ IC | ic concerns q or an attribute in R} ∪
{domain(r_i') = q' | $r_i \in R - T$} ∪
{range(r_i') = u_i | $r_i \in R - T$, range(r_i) = u_i} ∪
{q' ⊂ s_i | $t_i \in T$, range(t_i) = s_i}
{r_i' is single-valued (injective, total, surjective, domain-stable, range-stable) |
$r_i \in R - T$, r_i is single-valued (injective, total, surjective, domain-stable, range-stable)}

Transformation 6^{-1}:
Let CS = <L(P,F,C),IC> be a conceptual schema. The transformation below is applicable when P contains a type r and two types p and q such that r ⊂ p and r ⊂ q, and r is neither the domain nor the range of any attribute. The transformation is given by:

L(P,F,C) ↦ L(P - {r},F,C)
IC ↦ IC - {r ⊂ p, r ⊂ q}