A Methodology for the Design and Transformation of Conceptual Schemas

Christoph F. Eick
Department of Computer Science
Houston, TX 77204-3475, e-mail: ceick@cs.uh.edu

Abstract

Conceptual schemas have been recognized as an important tool for the design and evaluation of integrated databases and knowledge-based systems. The paper surveys the back end of a conceptual design methodology called ANNAPURNA. Its theoretical foundation that relies on multi-typed functional and existence dependencies is discussed. Quality measures for conceptual schemas are introduced. A general framework for the specification of conceptual schema transformations is proposed and algorithms for the evaluation and transformation of conceptual schemas are provided. The possibilities and limitations of using computerized tools for these tasks are discussed.

1. Introduction

In the last decade, *conceptual schemas* have been recognized as an important tool for the design and evolution of integrated databases and knowledge-based systems. Conceptual schemas describe [1] which classes of entities and propositions are of importance for a particular universe of discourse (UoD) of an application area. Furthermore, they specify which rule/constraints hold in the particular UoD.

Unfortunately, for a given UoD an unlimited number of potential conceptual schema candidates exists, which raises the questions "which of these candidates are *good* conceptual schemas", and "how can they be obtained in practice". Surprisingly, the first question, the evaluation of conceptual schemas - although important - has not obtained much attention by past research. In relational database theory [2, 3] quality is defined by the presence or absence of certain normal forms, which are defined in the context of special classes of rules, usually functional and multivalued dependencies. However, its definition of quality is somewhat restrictive in the sense that it relies on a two-valued definition of quality (a schema is either good or bad), and excludes other important criteria such as the complexity of a schema, or other classes of rules (e.g. subclass relationships) from the schema evaluation process. Even worse, quality measures have been largely ignored by research that directly focuses on conceptual schema design making conceptual schema design somewhat under-constrained in the sense that it has to come up with conceptual schemas, whose desired properties are only vaguely understood.

On the other hand, the second problem, the transformation of conceptual schemas, has been explored more systematically. Criteria have been introduced, to ensure that a schema transformation preserves the validity of the original schema [4, 5]. A number of special schema transformations have been explored in the literature to make conceptual schemas more "natural" [6, 7, 8, 9]. However, the lack of quality measures for conceptual data models makes the use of these transformations somewhat haphazard, because no clear understanding exists, how a good conceptual schema looks like.

We strongly believe that progress in conceptual schema design can only be made by approaches that are able to integrate quality measures and schema transformations systems relying on conceptual data models that have a sound theoretical foundation, which includes a clearly defined semantics and (possibly heuristic) inference rules to reason with conceptual schemas. The objective of this paper is to survey the back end of a conceptual schema design methodology, called *ANNAPURNA* (its front end has been described in [10]), that aims to automate conceptual schema design along the lines, outlined before, focussing on the transformation and evaluation of conceptual schemas. Before these subjects can be discussed, some introductory discussions are needed, which is the subject of the next section.

2. Valid and Good Conceptual Schemas

We mentioned in the introduction that many possible conceptual schemas exist for a given UoD. This raises the question, if a design procedure should consider any conceptual schema as a possible candidate. In the proposed methodology, we restrict our attention to conceptual schemas that are *valid*. Validity is defined with respect to the information requirements the data- or knowledge base to be designed has to satisfy. Two facets of validity are distinguished. The first facet is called *proposition completeness*. We call a conceptual schema proposition complete, if it allows the specification of proposition types that are to satisfy the information requirements. For example, if there are information requirements that request information about phone numbers and the conceptual schema does not define proposition types (e.g. contains the definition of an attribute to describe phone numbers) to make propositions concerning phone numbers, then the corresponding conceptual schema violates proposition completeness. The second facet of validity is called *rule correctness*. A conceptual schema is called rule correct,

if all the rules described in the conceptual schema hold in the universe of discourse (but our definition does not assume the reverse). For example, if the conceptual schema specifies that phone numbers are unique for employees, and it is possible that a current or future state of the UoD two employees share the same phone number, rule correctness is violated by the schema. If a conceptual schema is invalid, certain information requirements concerning the UoD cannot be satisfied by information bases under the conceptual schema, because of the impossibility to store the desired information. Obviously, any enhancement of a conceptual schema has to preserve validity; otherwise, it is useless. In the following, a transformation whose application preserves validity will be called *information preserving*. In general, we see one main objective of a conceptual schema in giving a good classification of the objects that appear in the universe of discourse of the application area. More specifically, in our approach, we consider a classification to be "good", if it

- facilitates the specification of rules that hold in the UoD. Ideally, if a data model of high expressive power is available, it should be possible to describe all rules that hold in a universe of discourse. However, in reality due to the limited expressiveness of most conceptual data models[1] this is not always true and it is at least questionable if it is even desirable. Assuming an approach that relies on data models of restricted expressive power, the percentage of the rules that hold in the UoD and are not expressed by the conceptual schema becomes an important factor, which we call the *expressiveness* of a conceptual schema.
- has a low *complexity* in terms of the number of classes and attributes needed for this particular specification.
- is normalized in the sense that objects that are structurally equivalent or similar are described in the same way in the conceptual schema. We call this virtue *normalizedness*.

Schema transformations are applied in the context of these virtues trying to improve the schema in one direction hopefully not losing too much with respect to the other virtues.

In the remainder of this paper we will try, using a non-trivial example, to make the characteristics of our approach, which were briefly outlined in this section, more transparent. First, the theoretical foundation of the conceptual data model underlying our methodology is introduced in section 3. Then, in section 4, the general framework how our methodology treats conceptual schema transformations will be discussed in some detail. Finally, section 5, focuses on the evaluation of conceptual schemas.

[1] This low expressiveness brings up other advantages such as complete axiomatization, inference rules, and the availability of automatic or semi-automatic tools that cannot be provided for more powerful data models due incomplete axiomatizations, decidability and complexity problems originated from the more complex class of propositions supported by these data models.

3. S-diagrams, Existence and Functional Dependencies

The objective of this section is to introduce the theoretical foundation that underlies our methodology. Our data model, which was influenced by the work on the binary relation model [11] and SDM [12], is called called *S-diagram* [10,13]. We will illustrate the problems of the transformation and evaluation of conceptual schemas using the following example of a hospital application.

```
schema HOSP1
class Patient
 attributes:
  p-name
   type: TEXT
class Hospital
 attributes:
  h-name
   property: unique
   type: TEXT
  in-city
   type: TEXT
class Treatment
 attributes:
  tr-patient
   type: Patient
  tr-name
   type: TEXT
  price
   type: FIXPOINT(9,2)
  tr-hosp
   property:optional, onto
   type: Hospital
  ins-nr
   property: optional
   type: INTEGER
  ins-name
   property: optional
   type: TEXT
```

The above schema deals with treatment of patients in and outside hospitals, whose main characteristics can be summarized as follows: Treatments have names (tr-name). Hospitals are located in cities and patients are usually insured. Hospital only treat insured patient, while uninsured patients might only be treated outside hospitals. A patient has at most one insurance. The same treatment has the same price in different hospitals.

Using S-diagrams it is possible to specify *classes*, *subclass connections* and *attributes*. An attribute has a *domain class* and a *range class* (for example, the attribute tr-patient has the domain class treatment and the range class Patient).

An attribute assigns to a member of the domain class zero, one or many members of the range class and each member of the range class may be attribute value zero, one or many times. The cardinality of an

attribute definition may be restricted using *labels* multivalued, unique, optional and onto:

- *multivalued* specifies that an attribute may have more than one value; otherwise, it it assumed to be single-valued.
- *optional* specifies that an attribute may have no value; otherwise, it is assumed to have at least one value.
- *onto* specifies that every member of the range class has to be referred at least once by the attribute.
- *unique* specifies that two distinct members of the domain class must have different values for the attribute.

In the example schema the attribute tr-hosp is labeled {optional,onto} expressing the following semantics: At most one hospital will perform a treatment (because there is no label multivalued), however, treatments may be performed outside of hospitals (because there is a label optional); every hospital has performed at least one treatment (because there is a label onto) and hospitals may perform several treatments (because there is no label unique).

An *S-diagram information base* contains description of *entities*. It stores extensions of unary predicates describing class memberships of entities, and binary predicates describing attributes of entities. For example, if an S-diagram information base contains $ins\text{-}nr(e, 12)$ this specifies that the entity e has the insurance number 12, or $Treatment(e)$ specifies that the entity e belongs to the class treatment. More precisely, we can define:

Definition: ib *is an information base under an S-diagram* S, if and only if:

(a) if ib contains $K(x)$, then K must be a class in S.
(b) if ib contains $att(y, z)$, then att must be an attribute in S.
(c) the assertions stored in ib observe the rules specified in S.

For example,

$ib_1 = \{Patient(1), p\text{-}name(1, Miller)\}$

is an information base under HOSP1, whereas ib_2 and ib_3

$$ib_2 = \{Hospital(3), Hospital(4), in\text{-}city(3, Houston), in\text{-}city(4, Udine), h\text{-}name(3, Marcus), h\text{-}name(4, Marcus)\}$$

$$ib_3 = \{Person(5), name(5, Jones)\}$$

are not information bases under HOSP1: ib_2 violates the rule that names of hospitals are unique, and ib_3 contains predicates *name* and *Person*, which are not defined in HOSP1.

As mentioned in the introduction, multiple possible conceptual schemas exist for the same UoD. An alternate, valid S-diagram HOSP2 for our hospital application is given below:

```
schema HOSP2
class Patient
 attributes:
  p-name
   type: TEXT
class Ins-patient
 subset of patient
 attributes
  ins-nr
   property: unique
   type: INTEGER
  ins-name
   type: integer
class Hospital
 attributes:
  h-name
   property: unique
   type: TEXT
  in-city
   type: TEXT
class Treatment
 attributes:
  tr-measure
   type: Measure
class Hosp-treatment
 subclass of Treatment
 attributes:
  tr-patient
   type: Ins-Patient
  tr-hosp
   property: onto
   type: Hospital
class Outside-treatment
 subclass of Treatment
 attributes:
  tr-patient
   type: Patient
class Measure
 attributes:
  tr-name
   property: unique
   type: TEXT
  price
   type: FIXPOINT(9,2)
```

The S-diagram, given above, uses a classification of a much finer degree of granularity, distinguishing between hospital treatments and outside treatments, between insured and insured patients, and considers measures to be objects (and no longer texts). Fig. 1 depicts the two S-diagrams HOSP1 and HOSP2 represented as a labeled directed graph, in which nodes represent classes, edges represent attributes, and edges labeled by 'S' represent subclass connections.

In general, S-diagrams are capable of expressing at-least-one constraints (in the following called *general existence dependencies* and represented by $\longmapsto$) and at-most-one constraints (called *general functional dependencies* and represented by $\longrightarrow$). The union of the two dependency classes is called *X-dependencies*. Existence dependencies express that the existence of certain attributes implies the existence of some other attributes. Functional dependencies express that if two entities agree in certain attributes, then they should also agree in certain other attributes. In this paper, we will introduce X-dependencies intuitively relying on examples (see [13], for a more formal treatment).

The following 2 dependencies are X-dependencies in our hospital application:

(1) ins-nr $\longmapsto$ ins-name

$(\forall x, y)(ins\text{-}nr(x, y) \Longrightarrow (\exists z) ins\text{-}name(x, z))$

meaning: If a patient has an insurance number, there must also be an insurance from which he has received his insurance number.

Remark: The above dependency is expressed in HOSP2 (but not in HOSP1), because both attributes ins-nr and ins-name are non-optional.

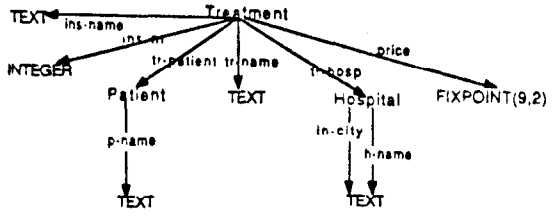
(2) ins-nr $\longrightarrow$ ins-name

$(\forall x, x', y, z, z')(ins\text{-}nr(x, y) \wedge ins\text{-}nr(x', y) \wedge ins\text{-}name(x, z) \wedge ins\text{-}name(x', z') \Rightarrow z = z')$

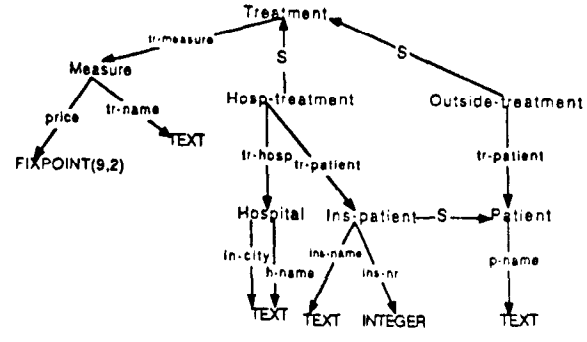
meaning: Insurance numbers are unique; different insurance-companies cannot use the same insurance-number.

Remark: The above dependency is expressed in HOSP2 by the fact that ins-nr is labeled unique and ins-name isn't labeled multivalued.

S-diagram are not only capable of representing dependencies between attributes belonging to the same class but also involving attributes of multiple classes. For formalizing relationships between multiple classes we introduce the notation of a *path*. Paths enable to describe relationships between entities belonging to different classes. The dependencies A $\overset{C}{\longmapsto}$ B and A $\overset{C}{\longrightarrow}$ B express existence/functional dependencies from the attributes/class-memberships contained in the set A to those in B connected using attributes contained in C.



S-Diagram HOSP1



S-Diagram HOSP2

Fig. 1: Two S-Diagrams for the same UoD

Let us clarify our path-concept by using the following example:

(3a) tr-name,in-city $\overset{tr-hospital}{\longrightarrow}$ price

$(\forall x, x', y, y', z, z', u, v)$
$(tr\text{-}name(x, u) \wedge tr\text{-}name(x', u) \wedge$
$in\text{-}city(y, v) \wedge in\text{-}city(y', v) \wedge$
$tr\text{-}hospital(x, y) \wedge tr\text{-}hospital(x', y') \wedge$
$price(x, z) \wedge price(x', z') \Rightarrow z = z')$

meaning: The same treatment has the same price in the same city.

Remark: The path $\{tr\text{-}hopital\}$ is used to specify a dependency that must hold for the members of the classes hospital and treatment, connecting treatment/hospital pairs (x and y, x' and y') that have to agree in their price attribute.

Due to the different classification the representation of the above dependency with respect to HOSP2 looks as follows:

(3b) tr-name,in-city $\overset{tr-measure,tr-hospital}{\longrightarrow}$ price

The above dependency is expressed in S-diagram HOSP2 because due to the fact that price isn't labeled multivalued and tr-name is labeled unique, a much stronger dependency tr-name $\longrightarrow$ price (each treatment has a unique price) holds in the UoD, which implies (3b). Interestingly, all three example dependencies are expressed in HOSP2, whereas none of them is expressed in HOSP1; that is, HOSP2 has a higher expressiveness.

The example raises two questions. First, can it be decided if (3b) is expressed in HOSP2? Second, how can HOSP2 be obtained starting from HOSP1? Section 4 will cope with the second problem, whereas sub-section 5.2 will give algorithms to cope with the first problem.

4. Conceptual Schema Enhancement

Our methodology provides a set of schema transformations that are analyzed in the context of the theoretical foundation, introduced in the last section. In general, in our approach a schema transformation is considered to be a 5-tupel (P,ST,ϕ,M,ρ), as depicted in Fig. 2, consisting of:

- A *precondition* P, which is a predicate that specifies under which circumstances the transformation preserves validity.
- A *structure transformation* ST that defines how the modified S-diagram can be received when the transformation is applied.
- A *information base transformation* ϕ that specifies how information bases defined under the original schema have to be transformed into information bases under the transformed schema.
- A *dependency mapping* M that maps dependencies defined in the context of the original schema into dependencies defined in the context of the transformed schema.

- A *reconstruction* ρ that specifies how information bases defined under the transformed schema can be transformed into information bases defined under the original schema.

4.1 Information Preserving Transformations

As mentioned before in our approach, validity is defined with respect to the satisfaction of information requirements, which can be considered as sets of queries Q defined on information bases under an S-diagram S. Intuitively, we call a transformation *information preserving*, if it transforms a valid S-diagram into a valid S-diagram. In the following, we use A → B to denote a function from A to B, and {A → B} to denote the set of functions from A to B. Furthermore, let:

SSS be the set of all possible S-diagrams

DDD be the set of all possible information bases

XXX be the set of all possible X-dependencies

S, S' be S-diagrams

IB_S be the set of all possible information bases under S

ib, ib' be information bases

W be an alphabet that is used to represent answers of queries

q, q' be queries. A query is considered to be a function ($\in \{IB_S \rightarrow W^*\}$) giving answers relative to the contents of an information base under a particular S-diagram.

Q is a set of queries

Q_S contains all possible queries for an S-diagram S

Relying on these notations, we can now define the terms S-diagram transformation and information preserving S-diagram transformation more precisely.

Definition: We call a 5-tuple T=(P,ST,ϕ,M,ρ) an *S-diagram transformation*, if and only if:

(i) P $\in$ {SSS → BOOLEAN}, ST $\in$ {SSS → SSS}, ϕ $\in$ {DDD → DDD}, D $\in \{2^{XXX} \rightarrow 2^{XXX}\}$, and ρ $\in$ {DDD → DDD}.

(ii) ϕ, ρ, ST and P are total functions.

(iii) $\forall S \in SSS\ \forall ib \in IB_S\ (P(S) \Rightarrow \phi(ib) \in IB_{ST(S)})$; that is, it is guaranteed that ϕ constructs information bases under the transformed S-diagram.

Definition: The application of a S-diagram transformation T is *information preserving with respect to a set of queries Q* for an S-diagram S, if and only if:

$$P(S) \Rightarrow (\forall q \in Q\ \exists q'\ \forall ib \in IB_S\ q(ib) = q'(\phi(ib)))$$

that is, for every query q a corresponding query q' can be found (constructively!), that produces the same results for information bases under the transformed schema. Fig. 3 depicts the conditions of the above definition — both queries q and q' have to produce the same results, represented by w in the figure.

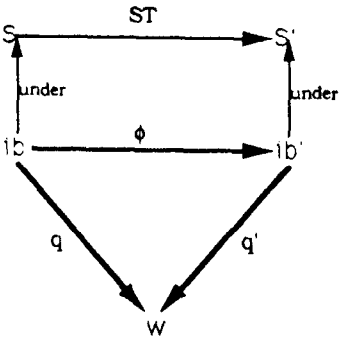


Fig. 3: Information Preserving Transformations

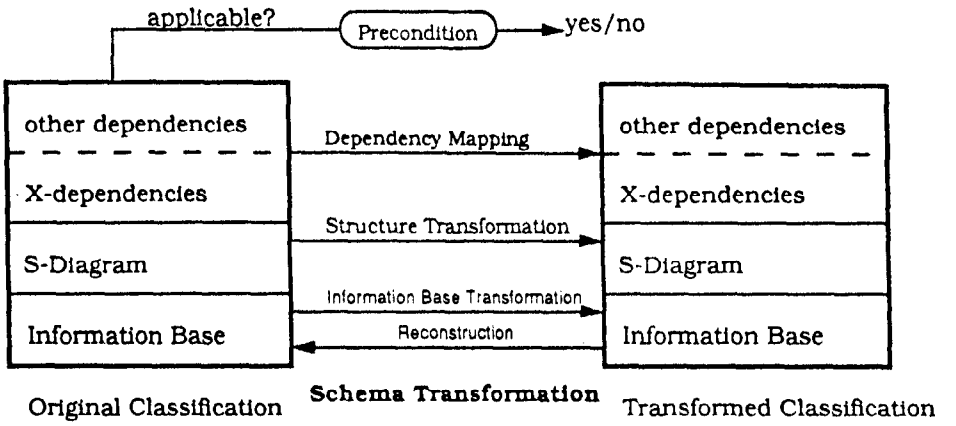


Fig. 2: S-Diagram Transformations.

In order to show that a transformation is information preserving, two approaches to construct q' for a given query $q \in Q$ seem to be attractive:

(1) A function QM $\in \{Q_S \rightarrow Q_{ST(S)}\}$ could be used to construct q' as follows:

$$q' = QM(q)$$

In other words, QM transforms a query defined in the context S into an equivalent query in the context of ST(S); we call this approach, *proof by query modification.*

(2) A function $\rho \in \{IB_{ST(S)} \rightarrow IB_S\}$ could be used, constructing q' as follows (o denotes the composition of functions):

$$q' = q \circ \rho$$

that is, in this solution an information base under the original schema is reconstructed and the original query q is used; we call this approach *proof by reconstruction.* Assuming the above framework, a transformation is trivially information preserving for an S-diagram S with respect to Q_S, if it is possible to reconstruct the original information base; that is, if

$$P(S) \Rightarrow (\forall ib \in IB_S \; ib = \rho(\phi(ib)))$$

holds.

If we compare our approach with other approaches, we can observe the following significant differences concerning which relationships between the transformed T(CS) and the original schema CS must hold, in order to apply a transformation T. In our framework, T(CS) and CS are considered to be "equivalent" if they are capable of satisfying a given set of information requirements. Our definition of validity is independent of the dependencies expressed by a conceptual schema. This contrasts other approaches [4, 5] that assume that two conceptual schemas are only equivalent, if they have the same expressive power in their underlying information bases and express the same dependencies. In our opinion, this definition of equivalence is too restrictive for conceptual schema design, because it excludes a number of useful transformations[2] and neglects the main purpose of data- and knowledge bases: the satisfaction of information requirements. Furthermore, defining schema equivalence with respect to the satisfaction of queries and not with respect to the contents of information bases (under a conceptual schema), allows us to abstract from the exact internal representation in a particular information base.

In the remainder of this section, the transformation system supported by our methodology is briefly introduced, and a single transformation is discussed in more detail.

[2] For example, those whose application expresses certain dependencies (not expressed so far) and loses others.

4.2 The Supported Transformation System

In detail, our transformations system consist of the following transformations, whose structure transformation is outlined below:

1) reverse(S,K,att)

semantics: a new S-diagram is generated by reversing the attribute att of the S-diagram S.

2) shift-down(S,att,att')

semantics: A new S-diagram is obtained by shifting the attribute att via the attribute att' in downward direction to the range class of att'.

example: this transformation was applied twice to HOSP1 when the attributes ins-nr and ins-name were moved along the path tr-patient to the class Patient.

3) decompose(S,K',att',K,A)

semantics: A new class K' is generated and connected by a new attribute att' to the class K; all attributes contained in the set A will be shifted to the class K'. For each value of the attribute A a new entity belonging to the class K' is generated.

example: this transformation has been applied to the class Treatment in HOSP1 relative to the attribute set $A = \{tr\text{-}name, price\}$ yielding a new class named Measure in HOSP2.

4) generalize(S,K,KSET,B)

semantics: A new class K is generated; all classes in KSET will become subclass of K and all attributes contained in the set B will be shifted to the new class K.

5) specialize(S,K',$\{(A_1, K_1), ..., (A_n, K_n)\}$)

semantics: n new subclasses $K_1, ..., K_n$ with attributes $A_1, ..., A_n$ of K generated. All attributes that belong to $\bigcup_{k=1}^{m} A_K$ will be removed from the class K.

example: The classes Hosp-treatment, Outside-treatment and Ins-patient of HOSP2 have been generated by specializing the classes Treatment and Patient of HOSP1.

6) chdom(S,att,K) *and* chrg(S,att,K)

semantics: Change of range/domain classes of attributes.

Additionally, inverse transformations of 2), 3), 4), and 5) are available.

One important use of the above transformation system is the formal specification of relationships between different conceptual schemas. For example, the following transformations describe the relationship between HOSP1 and HOSP2. HOSP2 can be received by applying the following sequence of transformations — in order to increase the readability, we introduced intermediate S-diagrams S1,...,S5.

S1=specialize(HOSP1, Treatment,
{(Hosp-treatment,{tr-hosp, ins-nr, ins-name, tr-patient}),
(Outside-treatment, {tr-patient})})

S2=shift-down(S1, ins-nr, tr-patient)

S3=shift-down(S2, ins-name, tr-patient)

S4=specialize(S3, Patient,
{(Ins-patient, {ins-nr, ins-name})})

S5=chdom(S4,Hosp-treatment.tr-patient, Ins-patient)

HOSP2=decompose(S5, Measure,
tr-measure, Treatment, {tr-name, price})

Here we will only discuss how the transformation *S2=shift-down(S1, ins-nr, tr-patient)* is handled in our approach, which is the subject of the next subsection.

4.3 A Closer Look to the Shift-down Transformation

The shift-down transformation moves an attribute (in the example ins-nr, a1 in theoretical discussions) from one class to another class along the path of an attribute (tr-patient in the example, att in theoretical discussions) — the insurance number is moved from the class Hosp-treatment to the class Patient. Fig. 4 depicts the structure transformation of shift-down.

It should be noted that the structure transformation alone is not sufficient to characterize a transformation completely, because it does not describe how the extensions of the modified attributes and classes have to be computed. In our approach, we specify information base transformations by using first order predicate calculus formulas that compute the extensions of modified or newly introduced classes and attributes. We use the suffix "'" to refer to classes and attributes of the transformed S-diagram, whereas class and attribute names without the suffix "'" refer to the original schema.

In the case of shift-down, only the attribute ins-nr is modified using the following information base transformation (IBT):

$$(IBT)\ a1'(x,y) := \exists z\ (a1(z,y)\ \wedge\ att(z,x))$$

In the example, the modified attribute ins-nr is computed as follows:

$$ins\text{-}nr'(x,y) := \exists z\ (ins\text{-}nr(z,y)\ \wedge\ tr\text{-}patient(z,x))$$

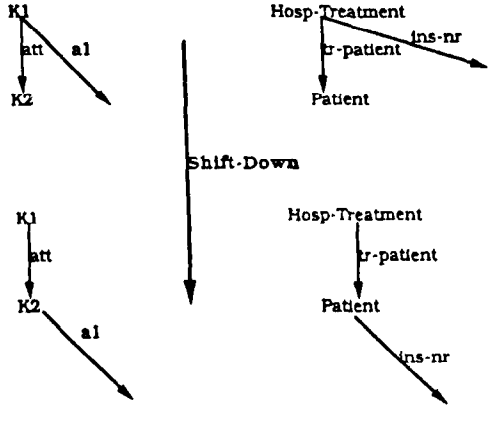


Fig. 4: Structure Transformation Shift-down

The shift-down transformation should only be applied, if the following two preconditions are satisfied; otherwise, it is not information preserving. An invalid S-diagram might be obtained, if one or both conditions are violated.

(p1) a1 $\longmapsto$ att

(p2) att is unique $\vee$
(a1 is non-optional $\wedge$ att $\longrightarrow$ a1)

(p1) states, that if an entity has the attribute a1 must have the attribute att, too. This requirement is quite obvious, because if it is violated, no instance of class K2 (patient in the concrete example) exists that can take the attribute a1; that is, the shift-down transformation, would lose the attribute information in the latter case.

If (p2) is violated it would be no longer possible to distinguish coverages by different insurances of different treatment of the same patient; e.g. if there are two treatments t1 and t2 of the same patient using different insurance numbers (e.g. different insurances cover the different treatments) it is no longer possible under the transformed S-diagram to distinguish which insurance covers which treatment.

In the example S-diagram S1 (p1) trivially holds, because tr-patient is not optional. Furthermore, in our special UoD, it holds that "patients have at most one insurance" (tr-patient $\longrightarrow$ ins-nr) and that the attribute ins-nr is non-optional, because "hospitals do not treat uninsured patients"; therefore, the application of the shift-down transformation is information preserving in this case.

In general, assuming that the transformation's preconditions are satisfied, it can be proven that the original information base can be reconstructed from a transformed information base without loss of information using the following reconstruction (RECON) for computing the values of the original ins-nr attribute:

$$(\text{RECON})\ ins\text{-}nr(x,y) := \exists z\ (tr\text{-}patient(x,z)\ \wedge\ ins\text{-}nr'(z,y))$$

That is, in the particular case of the shift-down transformation

$$P(S)\ \Rightarrow\ (\forall ib \in IB_S\ ib = \rho(\phi(ib)))$$

holds, in which ρ denotes the information base mapping expressed by (RECON), and ϕ denotes the information base mapping expressed by (IBT). That is, according to the definitions given in 4.1, the shift-down transformation is information preserving with respect to Q_S, if its preconditions (p1) and (p2) are satisfied.

The structure transformation, introduced so far, is still incomplete in the sense that it doesn't compute the labels of the transformed attribute a1'. Fortunately, the following simple rules can be used for the computation of the labels of a1':

(L1) a1' is non-optional $\Longleftrightarrow$ att is labeled onto $\wedge$ att $\longmapsto$ a1

(L2) a1' is labeled onto $\Longleftrightarrow$ a1 is labeled onto

(L3) a1' is multivalued $\Longleftrightarrow$ a1 is labeled multivalued

(L4) a1' is unique $\Longleftrightarrow$ a1 $\longrightarrow$ att

In our example, ins-nr' receives a label unique; all other labels remain unchanged.

When applying the shift-down transformation it is frequently possible to express certain dependencies between att and a1 that were not expressed in the original S-diagram, namely in the following two cases:

(g1) The transformation improves the quality by expressing att $\longrightarrow$ a1 in the transformed schema, if att is not labeled unique.

(g2) The transformation improves the quality by expressing att $\longmapsto$ a1 in the transformed schema, if a1 is labeled optional and att is labeled onto.

In the particular case, the conditions of (g1) are satisfied, tr-patient $\longrightarrow$ ins-nr holds and tr-patient is not unique. Therefore, the above dependency becomes expressed in the transformed schema. It should also be noted that tr-patient $\longrightarrow$ ins-nr was not expressed in the original schema, due to the fact that insurance numbers are unique for patients but not for treatments. That is, the quality is improved by expressing the dependency in the transformed schema.

In general, for each transformation we give a set of positive (and negative) cases, describing situations in which a particular transformation expresses dependencies that have not been expressed before (loses dependencies that have been expressed before).

Finally, the dependencies defined in the context of the original schema have to be redefined in the context of the transformed schema. In the case of shift-down, a relatively complex set of mapping rules has to be used to determine the transformed dependencies.

5. Evaluation of Conceptual Schema

In this section, we will discuss an evaluation function for conceptual schemas that incorperates the three virtues, outlined in section 2.

5.1 A Quality Measure for S-Diagrams

We assume that a UoD U and a set S-diagrams CAND={S_1,...,S_n} describing this UoD are given. Furthermore, for every schema candidate S_i ($0 < i < (n+1)$) a set X_i has been constructed which describes the X-dependencies that hold in the UoD. Some of the X-dependencies may be expressed in S_i, others may not.

The different schema candidates are evaluated using the following quality measure:

Let S_i be the S-diagram to be evaluated

qu_1 the number of functional dependencies (ϵX_i) that hold in U, but which are *not* expressed in S_i

qu_2 the number of existence dependencies(ϵX_i) that hold in U, but which are *not* expressed in S_i

qu_3 the number of attributes and subtype-connections in S_i

qu_4 the number of classes in S_i

qu_5 the number of labels of S_i

f be a function from N^5 to R^+ (N denotes the natural numbers and R^+ denotes the positive real numbers)

We define the *quality of S relative to U* as:

$$f(qu_1, qu_2, qu_3, qu_4, qu_5)$$

We say S_j is *better* than S_i, if f assigns a higher score to S_j than to S_i, and S_j is valid.

We assume that the following evaluation function f is used:

$$\frac{1}{3 * qu_1 + 3 * qu_2 + qu_3 + qu_4 + qu_5}$$

It is important to state that the values of f can be computed automatically: qu_3, qu_4 and qu_5 can be obtained easily, and an algorithm that decides if an X-dependency is expressed by an S-diagram is given in sub-section 5.2.

f incorporates three virtues. As we have seen section 3, general functional and existence dependencies are an upper bound for the expressive power of S-diagrams. Thus, the expressiveness of an S-diagram can be measured by the number of general functional and existence dependencies, that hold in U but which have not been expressed in S (qu_1 and qu_2 in f). The complexity of an S-diagram is measured by the number of classes, subtype-connections, and attributes of an S-diagram, expressed by qu_3 and qu_4.

Furthermore, our evaluation function incorporates a third virtue called *normalizedness*, which needs to be explained in more detail. Unfortunately, conceptual schema languages allow to describe the same proposition in different ways. It is desirable that propositions of structural similarity are described in the same way in the conceptual schema. We call this virtue of a conceptual schema *normalizedness*. In order to clarify this quality factor, let's consider the following example that concerns the representation of the treatment-patient relationship of the S-diagram HOSP1 ((1) in the Figure 5). However, this relationship could also be represented, by reversing the attribute direction, as depicted in (2) of Figure 5.

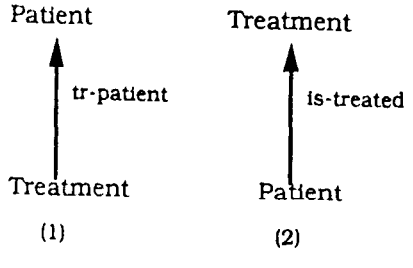


Fig. 5: 2 Similar S-Diagrams.

This situation, in which more than one construct is suitable to describe a part of a UoD adequately, occurs quite frequently in conceptual schema design processes. The quality factor of normalizedness should give a better evaluation to S-diagrams that uniquely use the *same* construct of the conceptual schema language, than to S-diagrams which use both representations (for some attributes the representation (1), for others (2)).

Violating the virtue of normalizedness has the following consequences:

- The conceptual schema is less understandable because similar objects are defined in a way which hides their similarity.
- One difficult task, when designing or extending a conceptual schema, is the detection of redundancies between different user views. If the same or similar objects are described in different ways, redundancies are much harder to detect.

Normalizedness is measured in the evaluation function using qu_5: S-diagrams that carry few labels are preferred. In S-diagrams having a low value for qu_5 the most general classes will be in or near the leaves of the S-diagram (relative to the shape of the arrow describing attribute and subtype relationships). The more specific classes have been defined by assuming the existence of lower level classes using aggregation or generalization [14]. For example in S-diagram HOSP2, which is highly normalized, the class Ins-patient has been defined by specializing the class Patient, and Hospital-treatment has been defined assuming the existence of the classes Hospital, Ins-patient and Treatment.

5.2 Expressed Dependencies of an S-Diagram

In this sub-section we will briefly introduce algorithms that decide if an X-dependency is expressed in an S-diagram. Two different algorithms are used for existence dependencies and functional dependencies. In order to discuss the two algorithms, we first have to introduce some notations:

Let
Att be an attribute, and K and K' be classes, then
dom(Att) denotes the domain class of Att
rg(Att) denotes the range class of Att
$K \sqsubseteq K'$ expresses that K is a subclass of K'
A^{-1} denotes the attribute received by reversing the attribute A

Furthermore, the rules specified below assume that
X,Y are sets containing attribute and class names
Z is a set containing attribute names
A is an attribute with dom(A)=K1 and rg(A)=K2

The algorithm that tests if a given existence dependency x is expressed in an S-diagram S is based on the following two rules.

(e1) A is not labeled optional $\Longrightarrow$
$(X \overset{Z}{\longmapsto} Y \cup \{K1\} \Longleftrightarrow X \overset{Z}{\longmapsto} Y \cup \{A\})$
A is labeled onto $\Longrightarrow$
$(X \overset{Z}{\longmapsto} Y \cup \{K2\} \Longleftrightarrow X \overset{Z}{\longmapsto} Y \cup \{A^{-1}\})$

(e2) $(((\exists A \in X)\ dom(A) \sqsubseteq K) \vee ((\exists C \in Z)\ dom(C) \sqsubseteq K \vee rg(C) \sqsubseteq K))) \Longrightarrow$
$(X \overset{Z}{\longmapsto} Y \cup K \Longleftrightarrow X \overset{Z}{\longmapsto} Y)$

Our algorithm consists of two steps. In a first step we try to weaken x using the rule (e1) by iteratively substituting class symbols for attribute symbols in the right hand side of x, as long as the rule (e1) is applicable. As a result of the first step, we receive a modified existence dependency x' which has been constructed so that:
(1) x $\Longrightarrow$ x'
(2) x' $\wedge$ "The constraints expressed by S" $\Longrightarrow$ x

If x' is always true, then x can be inferred from the constraints expressed by an S-diagram, which implies that x is expressed in S. Therefore, we try to show in a second step that x' is a tautology by applying the rule (e2). If it is possible to reduce the right side of x' to the empty set, then x is expressed in S, otherwise x is not expressed in S.

For example, if the above algorithm is applied to dependency (2)

ins-nr $\longmapsto$ ins-name

with respect to HOSP2, we would apply (E1) with $Y = \emptyset$ yielding:

ins-nr $\longmapsto$ Ins-patient

Note that Ins-patient is the domain class of the attribute ins-nr. Taking into consideration that $dom(ins\text{-}nr) = Ins\text{-}patient \sqsubseteq Ins\text{-}patient$ holds, we receive

ins-nr $\longmapsto \emptyset$

using (e2), which represents a tautology. Therefore, dependency (2) is expressed HOSP2.

The corresponding algorithm for functional dependencies uses the following four rules. Let A, K1, K2, X, Y, and Z be defined as before, and K be a class of S.

(f1) A is labeled unique $\Longrightarrow$
$(X \cup \{A, K1\} \overset{Z}{\longrightarrow} Y \Longleftrightarrow X \cup \{A\} \overset{Z}{\longrightarrow} Y)$
A isn't labeled multivalued $\Longrightarrow$
$(X \cup \{A^{-1}, K2\} \overset{Z}{\longrightarrow} Y \Longleftrightarrow X \cup \{A^{-1}\} \overset{Z}{\longrightarrow} Y)$

(f2) A is labeled unique $\Longrightarrow$
$(X \cup \{K1, K2\} \overset{Z \cup \{A\}}{\longrightarrow} Y \Longleftrightarrow X \cup \{K2\} \overset{Z \cup \{A\}}{\longrightarrow} Y)$
A isn't labeled multivalued $\Longrightarrow$
$(X \cup \{K1, K2\} \overset{Z \cup \{A\}}{\longrightarrow} Y \Longleftrightarrow X \cup \{K1\} \overset{Z \cup \{A\}}{\longrightarrow} Y)$

(f3) A isn't labeled multivalued $\Longrightarrow$
$(X \cup \{K1\} \overset{Z}{\longrightarrow} Y \Longleftrightarrow X \cup \{K1\} \overset{Z}{\longrightarrow} Y \cup \{A\})$

A is labeled unique $\Longrightarrow$

$(X \cup \{K2\} \xrightarrow{Z} Y \Longleftrightarrow X \cup \{K2\} \xrightarrow{Z} Y \cup \{A^{-1}\})$

(f4) $\{K\} \cup X \xrightarrow{Z} \{K\} \cup Y \Longleftrightarrow \{K\} \cup X \xrightarrow{Z} Y$

In a first step, a functional dependency x' is constructed by weakening x by iteratively applying the rules (f1) and (f2), respectively. In this process, the left side of x is augmented by further class names. Again, x is expressed in S, if x' is a tautology. Therefore, in a second step, the algorithm tries to reduce the right side of x' to the empty set by applying the rules (f3) and (f4), respectively.

If the above algorithm is applied to the dependency (3b), defined before in the context of HOSP2,

$$\text{tr-name,in-city} \xrightarrow{\text{tr-measure,tr-hospital}} \text{price}$$

we can weaken the above dependency taking advantage of the fact that tr-name is labeled unique yielding:

$$\text{tr-name,in-city,Measure} \xrightarrow{\text{tr-measure,tr-hospital}} \text{price}$$

Furthermore, taking advantage of the fact that price doesn't carry the label multivalued, we can apply (f3) recieving

$$\text{tr-name,in-city,Measure} \xrightarrow{\text{tr-measure,tr-hospital}} \emptyset,$$

which implies that the dependency (3b) is expressed in HOSP2.

6. Conclusion

This paper discussed the features of a methodology for enhancing conceptual schemas called ANNAPURNA. Our methodology is unique in the sense that we describe the semantics of schema transformations using a multi-typed approach to express data semantics relying on unary relations (classes), binary relations (attributes), and paths, rather than on a single-typed universal relation.

Quality measures for conceptual schemas were provided, and a general framework for studying schema transformations was introduces, that considers schema transformations to consist of a precondition, a structure transformation, an information base transformation, a reconstruction, and a dependency mapping. We illustrated our approach by discussing a transformation called shift-down, which has not been discussed in the literature before, in some detail. We claim, that this general framework, depicted in Fig. 2, is useful to study schema transformations systematically even in the context of other methodologies and/or other dependency classes. Furthermore, the validity of a conceptual schema is defined in our approach with respect to the satisfaction of information requirements. The notion of an information preserving transformation was introduced that that preserves the validity of the conceptual schema to be transformed. We demonstrated that this framework – compared with other approaches to define schema equivalence – is less restrictive and more flexible to cope with conceptual schema transformations.

In general, the computations needed for transforming and evaluating S-diagrams can be automated using a computerized tool. Although this is the case, we do not think that that it is desirable to use completely automatic design tools without human assistance. There are several reasons, why the cooperation of a human expert with a design tool is still necessary. First, if applied in practice the enhancement of conceptual schemas has to cope with problems such as missing or erroneous dependencies. In general, it cannot be inferred by a design tool which rules do or do not hold in a UoD. Therefore, dependencies can only be acquired by design tools by asking human experts, which frequently leads to misunderstandings and errors resulting in erroneous or incomplete specifications. The second reason is that finding a "the best" or even only a "good" conceptual schema is a highly complex search process, which requires a lot of heuristic knowledge to be computationally feasible. Again, the cooperation with a human expert seems to be necessary.

7. References

[1] ISO/TC97/SC5/WG3. Concepts and Terminology for the Conceptual Schema and the Information Base. *Publication number ISO/TC97/SC5/N695* (1982).

[2] D.Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.

[3] J. Ullman. *Principles of Data Base Systems.* Computer Science Press, Maryland, 1980.

[4] S. Jajodia, P. Ng, and F. Springsteel. The Problem of Equivalence for Entity-Relationship Diagrams. *IEEE Transactions on Software Engineering* (Sept. 1983).

[5] J. Grant. Constraint preserving and lossless database transformations. *Information Systems* 9 (2) 139-146 (1984).

[6] C. Batini, and G. Di Battista. A Methodology for Conceptual Documentation and Maintenance. *Informations Systems* 13 (3), 297-320 (1988).

[7] S. Ceri. *Methodology and tools for data base design.* North Holland, Amsterdam, 1983.

[8] O. Oren. A Method for the Optimization of a Conceptual Model. *Proc. First Int. Conf. on Data Engineering* (1984).

[9] D. Vermeir, and G. Nijssen. A procedure to define the object type structure of a conceptual schema. *Information Systems* 7 (3) 329-336 (1982).

[10] C.F. Eick, and P.C. Lockemann. Acquisition of Terminological Knowledge using Database Design Techniques. *Proc. ACM-SIGMOD Conference on Management of Data* 84-94 (1985).

[11] J.R. Abrial. Data Semantics. *Proc. IFIP TC2 Conference*, North Holland, Amsterdam, 1974.

[12] M. Hammer, and D. McLeod. Database Description with SDM: A Semantic Database Model. *ACM TODS* 6 (3) 351-386 (1981).

[13] C.F. Eick, and T. Raupp. Towards a Formal Semantics and Inference Rules for Conceptual Data Models. to appear in *Data & Knowledge Engineering* in 1991.

[14] J. Smith, and D. Smith. Data Abstractions – Aggregation and Generalization. *ACM TODS* 2 (1) 105-133 (1977).