Correct Schema Transformations[*]

Xiaolei Qian
Computer Science Laboratory
SRI International
qian@csl.sri.com

Abstract

We develop a formal basis of correct schema transformations. Schemas are formalized as abstract data types, and correct schema transformations are formalized as information-preserving signature interpretations. Our formalism captures transformations of all schema components, making it possible to transform uniformly constraints and queries along with structures. In addition, our formalism captures schema transformations between different data models as easily as those within the same data model. Compared with Hull's notion of relative information capacity, our notion of information preservation captures more schema transformations that are natural, and fewer schema transformations that are unnatural. Our work lays the foundation of a transformational framework of schema manipulations.

1 Introduction

Schema transformations, such as removing anomalies and redundancies, schema restructuring, and translating one schema to another (possibly in a different data model), constitute the major activities in schema integration [3]. Central to schema transformations is the need to compare the information content of schemas: in transforming one schema to another, the information content of the source schema should be preserved. A schema usually consists of three components [19]: (1) a set of structures, (2) a set of constraints on the structures, and (3) a set of operators on the structures. The constraints capture the invariant properties common to all instances of the schema, and the operators provide the vocabulary for formulating queries in the schema.

Most work on schema transformations has focused on the structure component of a schema (see [19, Chapter 14] for a summary of existing approaches). A classic example of schema transformations is the decomposition of a relation with a lossless join that preserves functional dependencies [20, Section 7.8]. When constraints are limited to functional dependencies in the relational model and schema transformations are expressed as conjunctive queries, Klug gave an algorithm for deriving constraints in the target schema from those in the source schema [13]. When constraints

[*]This work was supported in part by U.S. Department of Defense Advanced Research Projects Agency and U.S. Air Force Rome Laboratory under contracts F30602-92-C-0140 and F30602-94-C-0198, and in part by the National Science Foundation under grant ECS-94-22688.

are limited to functional and inclusion dependencies in the relational model, optimization rules are given in [7, 8] to remove redundant relations, attributes, and constraints in a schema. However, the general problem of constraint and operator transformations remains open. Since constraints capture important information in a schema, schema transformations without the constraint component are incomplete at best. Without the operator component, it is impossible to transform queries in the source schema to those in the target schema, even if the structure component is transformed correctly.

Another major problem with existing work is that schema transformations are almost always limited to schemas within the same data model—in most cases the relational model or some dialect of the entity-relationship model. For example, most approaches to schema integration make the assumption that the schemas to be integrated are already formulated in some canonical data model (see [3] for a survey). As pointed out in [19], schema transformations between schemas in different data models are usually specified by ad hoc mapping rules, with no formalisms to guarantee correctness in terms of either information preservation or equivalence. In [14], correct translation is studied from entity-relationship schemas to relational schemas, using a notion of schema equivalence from [11]. However, only primitive forms of cardinality constraints are translated, and query translation is not considered.

It is customary to treat a schema as a logical theory and instances of the schema as models of the theory [16]. What constitutes the information content of a schema? According to Bar-Hillel and Carnap [2], the (semantic) information carried by a logical theory (schema) can be characterized by the set of sentences (i.e., constraints) logically implied by the axioms of the theory relative to some language system (i.e., structures and operators). Thus, all three components of a schema contribute to its information content. In turn, the information content of the structure component of a schema can be characterized by the set of instances of the schema. In other words, a schema is capable of storing information not only in its instances but also in its structures, constraints, and operators.

Most work on schema transformations uses ad hoc and very limited measures in comparing the information content of schemas. In [6] for example, a schema contains less information than another if every instance of the first is an instance of the second. The first systematic study was done by Hull [11], where he introduced the notion of relative information capacity for comparing the information content of relational schemas. It has been used as the formal basis of schema containment and schema equivalence for object models [1, 12], and for entity-relationship models [15, 17]. In [21], Hull's notion is extended to take into account update semantics. However, the constraint and operator components of schemas are largely ignored in these studies, even though Hull observed that the structure component alone of a relational schema does not contain much information at all [11]. In fact, it is very difficult to generalize the notion of relative information capacity to deal with constraints and operators. For example, it is shown in [15] that the information capacity of a schema can be increased by reducing its capacity of storing constraints; and it is claimed in [17] that the information capacity of a schema remains the same even if we add new operators (surrogate key attributes).

Traditionally, schemas have been formalized using a structural approach, which describes the structure component of a schema using a (fixed) set of type constructors. However, the type constructors have been limited to simple ones that can be stored and manipulated directly, such as tuples and sets. Recently, Beeri [4, 5] has advocated a behavioral approach based on the theory of abstract data type (ADT) and algebraic specification (see [22] for a survey), which describes uniformly all components of a schema using a set of ADT specifications. The behavioral approach has the additional advantage of supporting more complex and extensible data models.

We adopt the behavioral approach to schema specification, and develop a formal basis of correct schema transformations that solves the problems discussed above. In particular, schemas are formalized as ADTs (Section 2), schema transformations are formalized as signature interpretations (Section 3), and correct schema transformations are formalized as information-preserving signature interpretations (Section 4). In Section 5, correct schema transformations are further generalized to support hidden symbols. We then compare our notion of information preservation with Hull's notion of relative information capacity (Section 6), and solve an open problem posted in [11]. Our formalism is used in Section 7 to show the correctness of common schema transformations that have been proposed in the literature, such as lossless join decomposition, redundancy removal, schema integration, and schema translation. A special kind of schema transformations, namely schema refinements, can also be used to describe efficient schema implementations. Finally, Section 8 provides some concluding remarks.

2 Schemas

2.1 Definitions

We borrow the definitions of order-sorted signatures and algebras from the standard abstract data type literature [10]. A ***partially ordered set***, or ***poset***, is a pair $\langle S, \leq \rangle$, where S is a set and $\leq$ is a partial ordering on S. Let S^* be the set of finite sequences of members in S. The partial ordering $\leq$ can be extended to S^* such that $\langle s_1 \dots s_n \rangle \leq \langle s'_1 \dots s'_n \rangle$ iff $s_i \leq s'_i$ for $1 \leq i \leq n$.

An ***order-sorted signature*** Σ is a triple $\langle S, \leq, \Omega \rangle$, where S is a set of ***sort symbols***, $\langle S, \leq \rangle$ is a poset, and Ω is a family of finite sets $\{\Omega_{v,s}\}_{v \in S^*, s \in S}$ of ***function symbols*** that satisfy the monotonicity condition

$$\Omega_{v,s} \cap \Omega_{v',s'} \neq \emptyset \text{ and } v \leq v' \text{ imply } s \leq s'.$$

We write $f\colon v \to s$ to denote $v \in S^*, s \in S$, and $f \in \Omega_{v,s}$. $\Omega_{\langle\rangle,s}$ is a set of ***constant symbols*** (i.e., 0-ary function symbols) of sort s. For example, we could have a signature BOOL containing one sort ***bool*** and two constant symbols ***true*** and ***false***. The logical connectives $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$ are considered as function symbols in BOOL. $\Omega_{v,bool}$ is a set of ***predicate symbols*** for $v \in S^*$. For every sort $s \in S$ we assume that there is an (infix) predicate symbol $=_s \in \Omega_{\langle s,s \rangle,bool}$. For BOOL, $=_{bool}$ is simply $\leftrightarrow$.

Given two signatures $\Sigma = \langle S, \leq, \Omega \rangle$ and $\Sigma' = \langle S', \leq', \Omega' \rangle$, we say that $\Sigma \subseteq \Sigma'$ if

1. $S \subseteq S'$,
2. $s_1 \leq s_2$ implies $s_1 \leq' s_2$ for $s_1, s_2 \in S$, and
3. $\Omega_{v,s} \subseteq \Omega'_{v,s}$ for $v \in S^*, s \in S$.

We say that Σ and Σ' are *union-compatible* if

1. $\leq$ is identical to $\leq'$ on $S \cap S'$, and
2. $\Omega_{v,s} \cap \Omega'_{v',s'} \neq \emptyset$ implies $v = v'$ and $s = s'$.

For signature $\Sigma = \langle S, \leq, \Omega \rangle$, the ***Σ-terms*** are defined inductively as the well-sorted composition of sorted variables, function symbols, and sorted quantifiers $\forall$ and $\exists$ in Ω. A ***Σ-formula*** is a Σ-term of sort *bool*. A ***Σ-sentence*** is a closed Σ-formula.

Let $\Sigma = \langle S, \leq, \Omega \rangle$ be a signature. An ***order-sorted Σ-algebra*** A consists of an S-indexed family of ***carrier sets*** $\{A_s\}_{s \in S}$, and a function $f_A: A_v \to A_s$ for every $\Omega_{v,s}$ in Ω and $f \in \Omega_{v,s}$, where $v = \langle v_1, \ldots, v_n \rangle$ and $A_v = A_{v_1} \times \cdots \times A_{v_n}$, such that

1. $s \leq s'$ implies $A_s \subseteq A_{s'}$, and
2. $f \in \Omega_{v,s} \cap \Omega_{v',s'}$ and $v \leq v'$ imply $f_A: A_v \to A_s$ equals to $f_A: A_{v'} \to A_{s'}$ on A_v.

A ***schema*** ? is a pair $\langle \Sigma, \Phi \rangle$, where $\Sigma = \langle S, \leq, \Omega \rangle$ is a signature and Φ is a set of Σ-sentences called ***axioms***. For every sort $s \in S$ we assume that the equality axioms of reflexivity, symmetry, transitivity, and substitutivity are in Φ. A ***?-instance*** is a Σ-algebra that satisfies Φ. The semantics of ? is given by the set of ?-instances. A Σ-sentence p is a ***?-constraint***, denoted as ? $\models p$, if p is a logical consequence of the ?-axioms. The set of ?-constraints forms ***?-theory***. A ***?-query*** is a Σ-formula.

2.2 Examples

For a schema ? $= \langle \Sigma, \Phi \rangle$, its signature Σ specifies the structure and operator components of ?, while its axioms Φ specify the constraint component of ?. A typical example is the specification of schema SET, which takes ATOM as a parameter schema with sort *atom*.

$$
\begin{array}{lll}
\text{SET(ATOM)} \overset{\text{def}}{=} (\textbf{sort} & \mathit{set}(\mathit{atom}) \\
\quad\quad \textbf{subsort} & \mathit{atom} \leq \mathit{set}(\mathit{atom}) \\
\quad\quad \textbf{function} & \{\}: \to \mathit{set} \\
 & \_ \circ \_ : \mathit{atom}, \mathit{set} \to \mathit{set} \\
 & \ldots \\
\quad\quad \textbf{axiom} & \neg(x \circ S = \{\}) \\
 & x \circ (y \circ S) = y \circ (x \circ S) \\
 & x \circ (x \circ S) = x \circ S \\
 & \ldots).
\end{array}
$$

The subsort declaration states that every atom is a set (of size one). There is a constant $\{\}$ denoting the empty set, and an insertion function $\circ$ for inserting an atom into a set. As a second example, LIBRARY1 is a schema specification in the object model.

$$
\begin{array}{ll}
\text{BOOK} \overset{\text{def}}{=} (\textbf{extend} & \text{STRING} \\
\quad\quad \textbf{sort} & \mathit{book} \\
\quad\quad \textbf{function} & \mathit{title}(\_): \mathit{book} \to \mathit{string}
\end{array}
$$

$$
\begin{array}{lll}
 & & \mathit{isbn}(\_)\colon \mathit{book} \rightarrow \mathit{string} \\
 & \textbf{axiom} & \mathit{isbn}(x) = \mathit{isbn}(y) \rightarrow x = y) \\
\text{AUTHOR} & \overset{\text{def}}{=} (\textbf{extend} & \text{STRING, INTEGER} \\
 & \textbf{sort} & \mathit{author} \\
 & \textbf{function} & \mathit{name}(\_)\colon \mathit{author} \rightarrow \mathit{string} \\
 & & \mathit{year\text{-}of\text{-}birth}(\_)\colon \mathit{author} \rightarrow \mathit{integer}) \\
\text{LIBRARY1} & \overset{\text{def}}{=} (\textbf{extend} & \text{BOOK, AUTHOR}).
\end{array}
$$

The keyword **extend** introduces a list of imported schemas. There are two classes of objects: books and authors. Books have attributes title and isbn, and authors have attributes name and year-of-birth. In addition, books have unique isbn values. As a third example, LIBRARY2 is a more elaborate schema specification in the complex-object model.

$$
\begin{array}{lll}
\text{LIBRARY2} & \overset{\text{def}}{=} (\textbf{extend} & \text{LIBRARY1, SET(BOOK)} \\
 & \textbf{function} & \mathit{author\text{-}of}(\_)\colon \mathit{author} \rightarrow \mathit{set}(\mathit{book}) \\
 & \textbf{axiom} & \neg(\mathit{author\text{-}of}(x) = \{\}) \\
 & & (\exists x)(y \in \mathit{author\text{-}of}(x))).
\end{array}
$$

Compared with LIBRARY1, authors in LIBRARY2 have an additional set-valued attribute denoting the set of books that they are an author of. The two additional axioms state that every author has at least one book, and every book has at least one author. As a fourth example, LIBRARY3 is an alternative schema specification in the entity-relationship model.

$$
\begin{array}{lll}
\text{LIBRARY3} & \overset{\text{def}}{=} (\textbf{extend} & \text{BOOL, LIBRARY1} \\
 & \textbf{function} & \mathit{authorship}(\_,\_)\colon \mathit{book}, \mathit{author} \rightarrow \mathit{bool} \\
 & \textbf{axiom} & (\exists x)\mathit{authorship}(x, y) \\
 & & (\exists y)\mathit{authorship}(x, y)).
\end{array}
$$

Compared with LIBRARY1, LIBRARY3 has a many-to-many relationship relating books to their authors. The two additional axioms state cardinality constraints on the relationship: every author has at least one book and every book has at least one author.

3 Schema Transformation

Let $\Sigma = \langle S, \leq, \Omega \rangle$ be a signature. We denote by $\Sigma_{v,s}(x)$ the set of Σ-terms of sort s whose list of free variables is x of sorts v, where $v \in S^*$ and $s \in S$.

The following definition is based on the notion of theory interpretation in [9, Section 2.7]. Let $\Sigma' = \langle S', \leq', \Omega' \rangle$ be another signature. A *signature interpretation* $\sigma\colon \Sigma \rightarrow \Sigma'$ is a pair $\langle \delta, \theta \rangle$, where

1. $\delta\colon S \rightarrow S'$ is a map such that $s_1 \leq s_2$ implies $\delta(s_1) \leq' \delta(s_2)$, and

2. θ is a family of maps $\{\theta_{v,s}\colon \Omega_{v,s} \rightarrow \Sigma'_{\delta^*(v),\delta(s)}(x)\}_{v \in S^*, s \in S}$

where $\delta^*(\langle v_1, \ldots, v_n \rangle)$ denotes $\langle \delta(v_1), \ldots, \delta(v_n) \rangle$. We write $\sigma(s)$ for $\delta(s)$, $\sigma(v)$ for $\delta^*(v)$, and $\sigma(f)$ for $\theta_{v,s}(f)$. We can extend σ to Σ-formulas as follows. Given a Σ-formula p, $\sigma(p)$ denotes the Σ'-formula resulted from replacing every term $f(t)$ in p by $\theta_{v,s}(f)[\sigma(t)/x]$, where $f \in \Omega_{v,s}$ and $\theta_{v,s}\colon \Omega_{v,s} \to \Sigma'_{\delta^*(v),\delta(s)}(x)$.

It should be noticed that the notion of signature interpretation is more general than the notion of signature morphism commonly seen in the algebraic specification literature [22], in that a function symbol in the source signature can be mapped to an arbitrary term, not just a function symbol, in the target signature.[1]

Let $\Sigma = \langle S, \leq, \Omega \rangle$ and $\Sigma' = \langle S', \leq', \Omega' \rangle$ be two signatures, and $\sigma\colon \Sigma \to \Sigma'$ be a signature interpretation. A map M_σ from Σ'-algebras to Σ-algebras can be induced from σ as follows. Given a Σ'-algebra A', we construct a Σ-algebra A by assigning the value of $\sigma(s)$ in A' to s for every sort symbol $s \in S$ and assigning the value of $\sigma(f)$ in A' to f for every function symbol $f \in \Omega$.

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas. A ***schema transformation*** $\sigma\colon ? \to ?'$ is a signature interpretation $\sigma\colon \Sigma \to \Sigma'$. Since ?-constraints and ?-queries are Σ-formulas, σ transforms not only the structures and operators of ?, but also the constraints and queries of ?. By using the more general notion of signature interpretation, constructs in the source schema can be transformed to combinations of constructs in the target schema.

For example, a schema transformation from BOOK to AUTHOR could be

$$\{\textit{book} \mapsto \textit{author}, \textit{title} \mapsto \textit{name}(x), \textit{isbn} \mapsto \textit{int-to-str}(\textit{year-of-birth}(x))\} \tag{1}$$

where ***int-to-str*** is a unary function symbol in schema STRING that converts an integer to a string. The BOOK-axiom is transformed to an AUTHOR-sentence

$$\textit{int-to-str}(\textit{year-of-birth}(x)) = \textit{int-to-str}(\textit{year-of-birth}(y)) \to x = y.$$

As we argued in Section 1, all components and instances of a schema contribute to its information content. A schema transformation preserves the structures and operators of the source schema. However, it is not sufficient to preserve either the constraints or the instances of the source schema. For example, the above schema transformation does not preserve the constraints of BOOK, since the BOOK-axiom is not mapped to an AUTHOR-constraint. The inverse of the above schema transformation

$$\{\textit{author} \mapsto \textit{book}, \textit{name} \mapsto \textit{title}(x), \textit{year-of-birth} \mapsto \textit{str-to-int}(\textit{isbn}(x))\} \tag{2}$$

where ***str-to-int*** is a unary function symbol in schema STRING that converts a string to an integer, does not preserve the instances of AUTHOR, since an AUTHOR-instance containing two authors with the same name but different years of birth does not correspond to any BOOK-instance.

[1]For ease of presentation, our notion of signature interpretation is less general than the notion of theory interpretation in [9], in that a sort symbol in the source signature can only be mapped to a sort symbol in the target signature. It is not hard to generalize the notion of signature interpretation to allow a sort symbol to be mapped to an arbitrary sort expression.

4 Correct Schema Transformation

4.1 Constraint-Preserving Transformation

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas, $\sigma: ? \to ?'$ be a schema transformation, and M_σ be the induced map from Σ'-algebras to Σ-algebras. We say that σ is ***constraint-preserving*** if, for every $?$-axiom p, $\sigma(p)$ is a $?'$-constraint.

Theorem 1 *Schema transformation σ is constraint-preserving iff M_σ maps every $?'$-instance to a $?$-instance.*

Proof Suppose that M_σ maps every $?'$-instance to a $?$-instance, but σ is not constraint-preserving. There is a $?$-axiom $p \in \Phi$ where $?' \not\models \sigma(p)$. There is a $?'$-instance I' in which $\sigma(p)$ is not satisfied, or equivalently, $\neg\sigma(p)$ is satisfied. Hence $\sigma(\neg p)$ is satisfied in I'. Let $I = M_\sigma(I')$, which is a $?$-instance. By the way in which I is constructed from I' through M_σ, $\neg p$ is satisfied in I, a contradiction.

Suppose that σ is constraint-preserving, but there is a $?'$-instance I' such that $I = M_\sigma(I')$ is not a $?$-instance. Let p be the $?$-axiom that is not satisfied in I. Apparently $\sigma(p)$ is satisfied in I'. By the way in which I is constructed from I' through M_σ, p is satisfied in I, a contradiction. □

Constraint-preservation is relatively straightforward to prove: all we need to show is that a schema transformation preserves the axioms of the source schema.

Constraint-preservation is not sufficient to preserve the instances of the source schema, since there might be a $?$-instance I where $I \neq M_\sigma(I')$ for any $?'$-instance I'. An example is schema transformation (2) in Section 3.

The simpliest form of constraint-preserving schema transformation is schema extension, where the schema transformation consists of identity maps, and every $?$-axiom is a $?'$-constraint.

Corollary 2 *Schema extension is constraint-preserving.*

For the examples in Section 2.2, LIBRARY1 can in fact be transformed to LIBRARY2 and LIBRARY3 by schema extension. As a more complicated example of constraint-preserving schema transformation, suppose that we extend LIBRARY3 to LIBRARY4:

$$
\begin{array}{lll}
\text{LIBRARY4} \overset{\text{def}}{=} (\textbf{extend} & \text{LIBRARY3, SET(BOOK)} \\
\qquad\qquad\quad \textbf{function} & \textit{author-of}'(\_): \textit{author} \to \textit{set}(\textit{book}) \\
\qquad\qquad\quad \textbf{axiom} & \textit{author-of}'(x) = \{y | \textit{authorship}(y, x)\}).
\end{array}
$$

The schema transformation from LIBRARY2 to LIBRARY4, which maps *author-of* to *author-of′* and everything else to itself, is constraint-preserving, because the two LIBRARY2-axioms are mapped to two LIBRARY4-constraints

$$\neg(\textit{author-of}'(x) = \{\})$$
$$(\exists x)(y \in \textit{author-of}'(x))$$

which follow from the axioms of LIBRARY3 and LIBRARY4.

4.2 Instance-Preserving Transformation

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas, $\sigma: ? \to ?'$ be a schema transformation, and M_σ be the induced map from Σ'-algebras to Σ-algebras. We say that σ is ***instance-preserving*** if, for every $?$-instance I, there is a $?'$-instance I' such that $M_\sigma(I') = I$.

Theorem 3 *Schema transformation σ is instance-preserving, iff p is a $?$-constraint for every Σ-sentence p where $\sigma(p)$ is a $?'$-constraint.*

Proof Suppose that $? \models p$ for every Σ-sentence p where $?' \models \sigma(p)$, but σ is not instance-preserving. Let U be the set of Σ-algebras that are images of $?'$-instances under M_σ, and $I \notin U$ be a $?$-instance. Suppose that there is a Σ-sentence p in the theory of U such that p is not satisfied in I and hence $? \not\models p$. Since p is in the theory of U, p is satisfied in every Σ-algebra in U, and hence $\sigma(p)$ is satisfied in every $?'$-instance. This means that $?' \models \sigma(p)$, a contradiction. Therefore every $?$-instance is in U, meaning that σ is instance-preserving, again a contradiction.

Suppose that there is a Σ-sentence p where $?' \models \sigma(p)$ and $? \not\models p$, but σ is instance-preserving. There is a $?$-instance I in which p is not satisfied, or equivalently $\neg p$ is satisfied. Since σ is instance-preserving, there is a $?'$-instance I' such that $M_\sigma(I') = I$. By the way in which I is constructed from I' through M_σ, $\sigma(\neg p)$ is satisfied in I', or equivalently $\neg\sigma(p)$ is satisfied in I'. Hence $\sigma(p)$ is not satisfied in I', a contradiction. □

Theorem 3 tells us that, for a schema transformation to be instance-preserving, it is equivalent that its inverse preserves the constraints of the target schema. However, since the inverse of a schema transformation might not exist or might not be computable, Theorem 3 does not provide a straightforward way of proving instance preservation.

Theorem 4 *Schema transformation σ is instance-preserving iff σ maps every non-$?$-constraint to a non-$?'$-constraint.*

Proof Suppose that σ maps every non-$?$-constraint to a non-$?'$-constraint. For every Σ-sentence p where $?' \models \sigma(p)$, p has to be a $?$-constraint. Hence σ is instance-preserving according to Theorem 3.

Suppose that σ is instance-preserving. According to Theorem 3, p is a $?$-constraint for every Σ-sentence p where $?' \models \sigma(p)$. For every Σ-sentence p where $? \not\models p$, $\sigma(p)$ cannot be a $?'$-constraint. □

Theorem 4 tells us that, for a schema transformation to be instance-preserving, it is equivalent that it preserves the non-constraints of the source schema. However, since the set of non-constraints of a schema could be infinite, Theorem 4 does not provide a way of proving instance preservation either. In fact, there are no known ways of proving instance preservation.

Instance preservation is not sufficient to preserve the constraints of the source schema, since there might be a $?'$-instance I' where $M_\sigma(I') \neq I$ for any $?$-instance I, meaning that σ might not be constraint-preserving according to Theorem 1. An example is schema transformation (1) in Section 3.

The simpliest form of instance-preserving schema transformation is schema reduction, where the schema transformation consists of identity maps, and every $?'$-axiom is a $?$-constraint.

Corollary 5 *Schema reduction is instance-preserving.*

For example, suppose that we define LIBRARY5 as follows:

$$\textsc{library5} \overset{\text{def}}{=} (\begin{array}{ll} \textbf{extend} & \textsc{library3}, \textsc{set}(\textsc{book}) \\ \textbf{function} & \textit{author-of}'(\_)\colon \textit{author} \to \textit{set}(\textit{book}) \\ \textbf{axiom} & \textit{authorship}(y,x) \to y \in \textit{author-of}'(x)). \end{array}$$

The schema transformation from LIBRARY4 of Section 4.1 to LIBRARY5, which consists of identity maps, is instance-preserving because LIBRARY5-theory is weaker than LIBRARY4-theory.

4.3 Information-Preserving Transformation

As we have argued in Section 1, a correct schema transformation should preserve all components and instances of the source schema. Since schema transformations preserve structures and operators, constraint-preserving schema transformations preserve constraints, and instance-preserving schema transformations preserve instances, we have the following definition for correct schema transformations.

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas, and $\sigma\colon ? \to ?'$ be a schema transformation. We say that σ is ***information-preserving*** if it is both constraint-preserving and instance-preserving. If σ is information-preserving, then we say that $?$ is ***contained*** in $?'$, denoted by $? \sqsubseteq ?'$. If $? \sqsubseteq ?'$ and $?' \sqsubseteq ?$, then we say that $?$ and $?'$ are ***equivalent***, denoted by $? \equiv ?'$.

For example, suppose that we define LIBRARY6 as follows:

$$\textsc{library6} \overset{\text{def}}{=} (\begin{array}{ll} \textbf{extend} & \textsc{bool}, \textsc{library1}, \textsc{set}(\textsc{book}) \\ \textbf{function} & \textit{authorship}(\_,\_)\colon \textit{book}, \textit{author} \to \textit{bool} \\ & \textit{author-of}'(\_)\colon \textit{author} \to \textit{set}(\textit{book}) \\ \textbf{axiom} & \textit{authorship}(y,x) \leftrightarrow y \in \textit{author-of}'(x) \\ & \neg(\textit{author-of}'(x) = \{\}) \\ & (\exists x)(y \in \textit{author-of}'(x))). \end{array}$$

The schema transformations from LIBRARY4 of Section 4.1 to LIBRARY6 and back, which consist of identity maps, are both information-preserving, and hence LIBRARY4 $\equiv$ LIBRARY6.

The simpliest forms of information-preserving schema transformation are instance-preserving schema extension and constraint-preserving schema reduction.

Given a signature $\Sigma = \langle S, \leq, \Omega \rangle$ and a function symbol $f\colon v \to s$ where $v \in S^*, s \in S$, but $f \notin \Omega_{v,s}$, a definition of f in Σ is an atomic formula

$$f(x) = F$$

where $F \in \Sigma_{v,s}(x)$ is a Σ-term and x are free variables in F of sorts v. Given a schema $? = \langle \Sigma, \Phi \rangle$, we can form a schema $?' = \langle \Sigma', \Phi' \rangle$ where $\Sigma' = \Sigma \cup \{f\}$ and $\Phi' = \Phi \cup \{f(x) = F\}$. The schema extension from $?$ to $?'$ is a ***schema extension by definition***.

Theorem 6 *Schema extension by definition produces equivalent schemas.*

Proof Let σ be the schema extension from ? to ?$'$. Since every non-?-constraint is a non-?$'$-constraint, σ is instance-preserving according to Theorem 4. Hence σ is information-preserving and ? $\sqsubseteq$?$'$.

Let τ be the schema transformation from ?$'$ to ? which maps f to F and all other symbols to themselves. Since τ maps the ?$'$-axiom $f(x) = F$ to a tautology and all other ?$'$-axioms to themselves, τ is constraint-preserving according to Theorem 1. Since τ maps every non-?$'$-constraint to a non-?-constraint, τ is instance-preserving according to Theorem 4. Hence τ is information-preserving and ?$'$ $\sqsubseteq$?. □

For example, the extension of LIBRARY3 of Section 2.2 to LIBRARY4 in Section 4.1 is by definition. Hence LIBRARY3 $\equiv$ LIBRARY4.

5 Schemas with Hidden Symbols

A ***schema with hidden symbols*** ? is a triple $\langle \Sigma, \Pi, \Phi \rangle$, where $\langle \Sigma, \Phi \rangle$ is a schema and $\Pi \subseteq \Sigma$ is a signature. Symbols in Π are ***hidden*** and symbols in $\Sigma - \Pi$ are ***visible***. The semantics of a schema with hidden symbols is given by the set of restrictions of $\langle \Sigma, \Phi \rangle$-instances to $\Sigma - \Pi$. A ?-***query*** is a $(\Sigma - \Pi)$-formula.

For example, recall that author names do not have to be unique in the specification of LIBRARY1 of Section 2.2. We could create a surrogate key for AUTHOR objects using hidden symbols, which is not visible to users in querying.

$$\begin{array}{lll}
\text{AUTHOR}' \overset{\text{def}}{\equiv} & (\textbf{extend} & \text{SYMBOL, STRING, INTEGER} \\
& \textbf{sort} & \mathit{author} \\
& \textbf{function} & \mathit{id}(\_){:}\ \mathit{author} \to \mathit{symbol} \\
& & \mathit{name}(\_){:}\ \mathit{author} \to \mathit{string} \\
& & \mathit{year\text{-}of\text{-}birth}(\_){:}\ \mathit{author} \to \mathit{integer}) \\
& \textbf{hidden-sort} & \mathit{symbol} \\
& \textbf{hidden-function} & \mathit{id} \\
& \textbf{axiom} & \mathit{id}(x) = \mathit{id}(y) \to x = y).
\end{array}$$

Let ? $= \langle \Sigma, \Pi, \Phi \rangle$ and ?$'$ $= \langle \Sigma', \Pi', \Phi' \rangle$ be two schemas with hidden symbols. A schema transformation σ: ? $\to$?$'$ is a signature interpretation $\sigma: (\Sigma - \Pi) \to (\Sigma' - \Pi')$. The induced map M_σ is a map from $(\Sigma' - \Pi')$-algebras to $(\Sigma - \Pi)$-algebras.

For example, the schema transformations from AUTHOR to AUTHOR$'$ and back, which consist of identity maps, are both information-preserving. Hence AUTHOR $\equiv$ AUTHOR$'$. Notice that, contrary to the claim in [17], the two schemas would not be equivalent if the function symbol *id* were visible in AUTHOR$'$, because no schema transformations from AUTHOR$'$ to AUTHOR could map *id* in such a way that preserves the axiom of AUTHOR$'$.

Theorem 7 *Instance-preserving schema extension with hidden symbols produces equivalent schemas.*

Proof Suppose that $?$ is extended to $?'$ with hidden symbols by a instance-preserving schema transformation. Obviously $? \sqsubseteq ?'$. Since every $?'$-instance restricted to $\Sigma' - \Pi'$ is a $?$-instance, and every $?$-instance is a $?'$-instance restricted to $\Sigma' - \Pi'$ due to instance preservation, $?' \sqsubseteq ?$. □

6 Relationship to Hull's Notion

In [11], Hull introduced four progressively more restrictive measures of the information content of schemas. We first briefly define these measures of relative information capacity, and then characterize their relationships to our notion of information preservation. Through this, we solve an open problem posted in [11].

A schema $? = \langle \Sigma, \Phi \rangle$ is ***relational*** if every function symbol in Σ is either a constant or a predicate.[2] Without further notice, we assume that all schemas in this section are relational.

6.1 Relative Information Capacity

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas. The domain of a $?$-instance I, denoted as $\mathrm{dom}(I)$, is the union of its carrier sets. Let **DOM** be a set of values containing the domains of all $?$-instances and $?'$-instances, and $Z \subseteq \mathbf{DOM}$.

1. A Z-permutation of **DOM** is a bijective map on **DOM** that is identical on Z.
2. A map M from $?$-instances to $?'$-instances is Z-generic if it commutes with every Z-permutation of every $?$-instance.
3. A map M from $?$-instances to $?'$-instances is Z-internal if $\mathrm{dom}(M(I)) \subseteq \mathrm{dom}(I) \cup Z$ for every $?$-instance I.

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas. Also let M be a map from $?$-instances to $?'$-instances, and M' be a map from $?'$-instances to $?$-instances. $?'$ dominates $?$ via (M, M') if $M' \circ M$ is the identity map on $?$-instances.

1. $?'$ dominates $?$ ***absolutely***,[3] denoted as $? \preceq ?'(\mathrm{abs})$, if $?'$ dominates $?$ via (M, M') for some M, M'.
2. $?'$ dominates $?$ ***internally***, denoted as $? \preceq ?'(\mathrm{int})$, if there is a finite $Z \subseteq \mathbf{DOM}$ such that $?'$ dominates $?$ via (M, M') for some Z-internal M, M'.
3. $?'$ dominates $?$ ***generically***, denoted as $? \preceq ?'(\mathrm{gen})$, if there is a finite $Z \subseteq \mathbf{DOM}$ such that $?'$ dominates $?$ via (M, M') for some Z-generic M, M'.

[2]This is consistent with Hull's definition, in which calculus expressions could contain constants, and constraints could be more general than key dependencies (e.g., constants are constrained by unique name axioms).

[3]This definition is from [12], which is simpler and is not limited to relational schemas. The two definitions are equivalent for relational schemas [11].

4. $?'$ dominates $?$ ***calculously***, denoted as $? \preceq ?'$(calc), if there are two schema transformations $\sigma\colon ? \to ?'$ and $\tau\colon ?' \to ?$ with induced maps M_σ and M_τ respectively,[4] such that $?'$ dominates $?$ via (M_τ, M_σ).

Hull has shown that calculus dominance implies generic dominance, which in turn implies internal dominance, which in turn implies absolute dominance. Moreover, the implication from generic to internal dominance is strict if schemas contain certain kinds of constraints, namely key dependencies. However, it remains open whether the implications from calculus to generic dominance, and from internal to absolute dominance are strict.

6.2 Relationship to Relative Information Capacity

Let $? = \langle \Sigma, \Phi \rangle$ and $?' = \langle \Sigma', \Phi' \rangle$ be two schemas.

Theorem 8 *If $? \preceq ?'$(calc), then $? \sqsubseteq ?'$.*

Proof Let $\sigma\colon ? \to ?'$ and $\tau\colon ?' \to ?$ be two schema transformations with induced maps M_σ and M_τ respectively, such that $?'$ dominates $?$ via (M_τ, M_σ). By definition of calculus dominance, M_σ maps every $?'$-instance to a $?$-instance, and hence σ is constraint-preserving. For every $?$-instance I, $M_\tau(I)$ is a $?'$-instance such that $M_\sigma(M_\tau(I)) = I$. Hence σ is instance-preserving. Therefore σ is information-preserving. □

Theorem 9 *If $? \sqsubseteq ?'$, then $? \preceq ?'$(abs).*

Proof Let $\sigma\colon ? \to ?'$ be an information-preserving schema transformation with induced map M_σ. Since σ is constraint-preserving, M_σ maps every $?'$-instance to a $?$-instance. Since σ is instance-preserving, for every $?$-instance I there is a $?'$-instance I' such that $M_\sigma(I') = I$. Hence there is a map M from $?$-instances to $?'$-instances such that $M_\sigma \circ M$ is the identity map on $?$-instances. Therefore $?'$ dominates $?$ via (M, M_σ). □

There exist schemas $?$ and $?'$ such that $? \preceq ?'$(gen,int,abs) but $? \not\sqsubseteq ?'$ (and hence $? \not\preceq ?'$(calc)). For example, consider the following schemas.

SCHEMA1 $\overset{\text{def}}{=}$ (**sort** *city*
  axiom $(\exists x, y)\neg(x = y)$)
SCHEMA2 $\overset{\text{def}}{=}$ (**sort** *town*).

A SCHEMA1-instance is a set containing at least two cities, while a SCHEMA2-instance could be a singleton set. Hence SCHEMA1 is logically stronger than SCHEMA2. A map M from SCHEMA1-instances to SCHEMA2-instances could map every SCHEMA1-instance to the identical SCHEMA2-instance. On the other hand, a map M' from SCHEMA2-instances to SCHEMA1-instances could

[4]This definition is equivalent to Hull's definition, because relational calculus is equivalent to first-order predicate calculus, which is the language of relational schemas.

map every SCHEMA2-instance containing at least two towns to the identical SCHEMA1-instance; and map every singleton SCHEMA2-instance to the SCHEMA1-instance where *city* is assigned the set $\{\mathrm{SF}, \mathrm{LA}\}$. Since these two maps are both $\{\mathrm{SF}, \mathrm{LA}\}$-generic, and $M' \circ M$ is the identity map on SCHEMA1-instances, we conclude that SCHEMA1 $\preceq$ SCHEMA2(gen,int,abs). However, any schema transformation from SCHEMA1 to SCHEMA2 will have to map *city* to *town*. The only function symbol in SCHEMA1 is $=_{city}$. If it is mapped to $=_{town}$, then the image of the SCHEMA1-axiom is not a SCHEMA2-constraint. If it is mapped to an equivalence relation $\simeq$ other than $=_{town}$, then the image of the SCHEMA1-axiom cannot be true in any singleton SCHEMA2-instances. In either case, the schema transformation cannot be constraint-preserving, and hence SCHEMA1 $\not\sqsubseteq$ SCHEMA2.

This example tells us that Hull's notion could find dominance relationships between schemas that are not naturally related through dominance relationships: a schema should not be capable of storing more information if we reduce its capability of storing constraints. Our notion rules out such unnatural dominance relationships.

There also exist schemas ? and ?$'$ such that ? $\sqsubseteq$?$'$ (and hence ? $\preceq$?$'$(abs)) but ? $\not\preceq$?$'$(int,gen,calc). For example, consider the following schema.

SCHEMA3 $\overset{\text{def}}{=}$ (**extend** SYMBOL, STRING, BOOL
function $R_1(\_,\_)$: *symbol*, *string* $\rightarrow$ *bool*
axiom $R_1(x,y) \wedge R_1(x,z) \rightarrow y = z$)
SCHEMA4 $\overset{\text{def}}{=}$ (**extend** STRING, BOOL
function $R_2(\_)$: *string* $\rightarrow$ *bool*)

and a schema transformation from SCHEMA4 to SCHEMA3 that maps R_2 to the SCHEMA3-term $(\exists x)R_1(x,y)$. In other words, R_2 is the projection of R_1 on the non-key column. It is easy to show that the schema transformation is information-preserving, and hence SCHEMA4 $\sqsubseteq$ SCHEMA3. However, any map from a SCHEMA4-instance to a SCHEMA3-instance will have to invent as many distinct values as the size of R_2 for the key column of R_1. Since there are arbitrarily large SCHEMA4-instances, the map cannot be Z-internal for a fixed finite Z. So SCHEMA4 $\not\preceq$ SCHEMA3(int,gen,calc).

This example tells us that there are natural dominance relationships between schemas that are not captured by Hull's notion: the capability of storing information in a non-keyed schema should be increased by adding a key column. Our notion captures such natural dominance relationships.

The above examples also show that calculus dominance is strictly more restrictive than our notion, which is in turn strictly more restrictive than absolute dominance. Moreover, our notion is not comparable to generic or internal dominance. As a consequence, we have solved an open problem posted by Hull:

Corollary 10 *Calculus dominance is strictly more restrictive than generic dominance. Internal dominance is strictly more restrictive than absolute dominance.*

7 Applications

7.1 Lossless Join Decomposition

Consider Example 7.8 from [20, page 395], with the following two schemas:

$$\begin{array}{lll}
\textsc{schema5} \overset{\text{def}}{=} (\textbf{extend} & \textsc{string},\ \textsc{bool} \\
\qquad \textbf{function} & R(\_,\_,\_,\_)\colon \textit{string}, \textit{string}, \textit{string}, \textit{string} \to \textit{bool} \\
\qquad \textbf{axiom} & R(x,y,z,w) \wedge R(x,y',z',w') \to y = y' \\
 & R(x,y,z,w) \wedge R(x,y',z,w') \to w = w')
\end{array}$$

$$\begin{array}{lll}
\textsc{schema6} \overset{\text{def}}{=} (\textbf{extend} & \textsc{string},\ \textsc{bool} \\
\qquad \textbf{function} & R_1(\_,\_)\colon \textit{string}, \textit{string} \to \textit{bool} \\
 & R_2(\_,\_,\_)\colon \textit{string}, \textit{string}, \textit{string} \to \textit{bool} \\
\qquad \textbf{axiom} & R_1(x,y) \wedge R_1(x,y') \to y = y' \\
 & R_2(x,y,z) \wedge R_2(x,y,z') \to z = z').
\end{array}$$

A schema transformation from SCHEMA5 to SCHEMA6 could map R to the SCHEMA6-term $R_1(x,y) \wedge R_2(x,z,w)$. A schema transformation from SCHEMA6 to SCHEMA5 could map R_1 to the SCHEMA5-term $(\exists z,w)R(x,y,z,w)$ and map R_2 to the SCHEMA5-term $(\exists y)R(x,y,z,w)$. It is not hard to verify that SCHEMA5 $\equiv$ SCHEMA6. Since SCHEMA5 is not in BCNF while SCHEMA6 is, the latter is considered a better schema design.

7.2 Redundancy Removal

Given two schemas ? $= \langle \Sigma, \Phi \rangle$ and ?$' = \langle \Sigma', \Phi' \rangle$ where ? $\equiv$?$'$, We say that ?$'$ is less redundant than ? if Σ' contains fewer or simpler signature symbols than Σ, or Φ' contains weaker axioms than Φ. In this case, we could replace ? by ?$'$ to achieve a more concise representation of the same information. Consider the following schemas in the relational model:

$$\begin{array}{lll}
\textsc{schema7} \overset{\text{def}}{=} (\textbf{extend} & \textsc{string},\ \textsc{bool} \\
\qquad \textbf{function} & R_1(\_,\_,\_), R_2(\_,\_,\_)\colon \textit{string}, \textit{string}, \textit{string} \to \textit{bool} \\
\qquad \textbf{axiom} & R_1(x,y,z) \wedge R_1(x,y',z') \to y = y' \wedge z = z' \\
 & R_2(x,y,z) \wedge R_2(x,y',z') \to y = y' \wedge z = z' \\
 & R_1(x,y,z) \to (\exists y')R_2(x,y',z))
\end{array}$$

$$\begin{array}{lll}
\textsc{schema8} \overset{\text{def}}{=} (\textbf{extend} & \textsc{string},\ \textsc{bool} \\
\qquad \textbf{function} & R'_1(\_,\_)\colon \textit{string}, \textit{string} \to \textit{bool} \\
 & R'_2(\_,\_,\_)\colon \textit{string}, \textit{string}, \textit{string} \to \textit{bool} \\
\qquad \textbf{axiom} & R'_1(x,y) \wedge R'_1(x,y') \to y = y' \\
 & R'_2(x,y,z) \wedge R'_2(x,y',z') \to y = y' \wedge z = z' \\
 & R'_1(x,y) \to (\exists y',z')R'_2(x,y',z')).
\end{array}$$

A schema transformation from SCHEMA7 to SCHEMA8 could map R_1 to the SCHEMA7-term $R'_1(x,y) \wedge (\exists y')R'_2(x,y',z)$. A schema transformation from SCHEMA8 to SCHEMA7 could map R'_1 to the SCHEMA8-term $(\exists z)R_1(x,y,z)$. It is not hard to verify that SCHEMA7 $\equiv$ SCHEMA8. Compared to SCHEMA8, SCHEMA7 has more redundancy because the third column of R_2 is redundantly stored as the third column of R_1. Hence replacing SCHEMA7 by SCHEMA8 removes this redundancy. This schema transformation has been proposed in the literature [8, 17].

7.3 Schema Integration

Let $?_i = \langle \Sigma_i, \Phi_i \rangle$ for $1 \leq i \leq n$ be n schemas such that Σ_i and Σ_j are union-compatible for $1 \leq i, j \leq n$. When we integrate these schemas into one, the first step is to identify the semantic relationships between them. Suppose that the semantic relationships are expressed over Σ_i for $1 \leq i \leq n$, and possibly an additional signature Σ'; and consist of a set of $(\bigcup_{i=1}^{n} \Sigma_i \cup \Sigma')$-sentences Φ'. An integration of $?_i$, where $1 \leq i \leq n$, is a schema $? = \langle \Sigma, \Phi \rangle$ such that

$$\langle \bigcup_{i=1}^{n} \Sigma_i \cup \Sigma', \bigcup_{i=1}^{n} \Phi_i \cup \Phi' \rangle \equiv ? .$$

Thus, schema integration can be viewed as applying information-preserving schema transformations to the union of component schemas and their semantic relationships to improve the quality of the integrated schema (e.g., to remove redundancy). Consider a schema specification similar to LIBRARY2 of Section 2.2:

$$\begin{array}{lll}
\text{LIBRARY2}' \overset{\text{def}}{=} (\textbf{extend} & \text{LIBRARY1}, \text{SET}(\text{AUTHOR}) \\
\qquad\qquad\quad \textbf{function} & \textit{authored-by}(\_)\text{: } \textit{book} \rightarrow \textit{set}(\textit{author}) \\
\qquad\qquad\quad \textbf{axiom} & \neg(\textit{authored-by}(x) = \{\}) \\
 & (\exists x)(y \in \textit{authored-by}(x))).
\end{array}$$

Suppose that we integrate LIBRARY2 and LIBRARY2$'$. The semantic relationship between them can be expressed as

$$x \in \textit{author-of}(y) \leftrightarrow y \in \textit{authored-by}(x).$$

Intuitively, the two schemas contain the same information represented differently. Hence their union is redundant: the many-to-many relationship between authors and books is represented by two multi-valued maps that are inverses of each other. It is not hard to verify that the union of LIBRARY2 and LIBRARY2$'$ together with the above semantic relationship is equivalent to LIBRARY3 of Section 2.2. Hence LIBRARY3 can be taken as their integration. This schema transformation has been proposed in the literature [18].

7.4 Schema Translation

Given two schemas $?$ and $?'$, a schema transformation $\sigma\colon ? \rightarrow ?'$ is a translation from $?$ to $?'$ if $? \equiv ?'$.

Schema translation between different data models is not any more difficult than that within the same data model, because our schema formalism really blurs the difference between data models: data models differ in the data types they support. So we could have a schema specified in a mixture of data models, e.g., LIBRARY4 of Section 4.1 is in a combination of complex-object and entity-relationship models.

As an example of schema translation, suppose that we extend LIBRARY3 of Section 2.2 with an additional axiom $\textit{name}(x) = \textit{name}(y) \rightarrow x = y$, which can then be translated to the following schema in the relational model:

$$\begin{array}{lll}
\textsc{library3}' \overset{\text{def}}{=} (\textbf{extend} & \textsc{string}, \textsc{integer}, \textsc{bool} \\
\textbf{sort} & s_1, s_2 \\
\textbf{function} & R_1(\_,\_) \colon s_1, \textit{string} \to \textit{bool} \\
 & R_2(\_,\_) \colon s_2, \textit{integer} \to \textit{bool} \\
 & R(\_,\_) \colon s_1, s_2 \to \textit{bool} \\
\textbf{axiom} & R_1(x,y) \wedge R_1(x,z) \to y = z \\
 & R_2(x,y) \wedge R_2(x,z) \to y = z \\
 & (\exists y) R_1(x,y) \\
 & (\exists y) R_2(x,y))
\end{array}$$

through the schema transformation

$$\left\{ \begin{array}{l}
\textit{book} \mapsto s_1, \textit{isbn} \mapsto x, \textit{title} \mapsto \{y | R_1(x,y)\}, \\
\textit{author} \mapsto s_2, \textit{name} \mapsto x, \textit{year-of-birth} \mapsto \{y | R_2(x,y)\}, \\
\textit{authorship} \mapsto R(x,y)
\end{array} \right\}.$$

Notice that, because of the LIBRARY3′-axioms, the images of *title* and *year-of-birth* are always total and evaluate to singleton sets. Also notice that the inclusion dependencies from the two columns of R to the first columns of R_1 and R_2 respectively are satisfied automatically through sort constraints and the last two LIBRARY3′-axioms. Similar schema translations from the entity-relationship model to the relational model have been proposed in [14].

7.5 Schema Refinement

Given two schemas ? and ?′, a schema transformation σ: ? $\to$?′ is a refinement of ? to ?′ if ? $\sqsubseteq$?′. Thus schema refinements can increase the information content of the schemas being refined.

Advanced data models provide many powerful abstractions that make schema specifications concise and semantically clear. However, the added expressive power comes with a price: there is no known method to infer simple and efficient storage implementations for these abstractions [5], because many abstractions no longer have a single implementation that is tolerably efficient over the entire query space. For example, a set-valued attribute can be implemented by a relation, a list, a bit-vector, etc.. Such implementation decisions can be encoded as schema refinements. When guided by the characteristics of typical queries, schema refinements can be applied to derive efficient implementations of schemas.

As an example, consider the following specification of schema SEQ:

$$\begin{array}{lll}
\textsc{seq}(\textsc{atom}) \overset{\text{def}}{=} (\textbf{sort} & \textit{seq}(\textit{atom}) \\
\textbf{subsort} & \textit{atom} \le \textit{seq}(\textit{atom}) \\
\textbf{function} & [\,] \colon \to \textit{seq} \\
 & \_ \bullet \_ \colon \textit{atom}, \textit{seq} \to \textit{seq} \\
 & \ldots \\
\textbf{axiom} & \neg(x \bullet S = [\,]) \\
 & x \bullet S = x' \bullet S' \to x = x' \wedge S = S' \\
 & \ldots).
\end{array}$$

We could imagine that SEQ has a range equality predicate $\_ =_r \_$: *seq*, *seq* $\rightarrow$ *bool*, such that two sequences are range-equal iff they contain the same set of elements. It is not hard to show that the schema transformation from SET of Section 2.1 to SEQ

$$\{set \mapsto seq, \{\} \mapsto [\,], \circ \mapsto x \bullet y, = \mapsto x =_r y\}$$

is information-preserving. Now consider LIBRARY$2'$ of Section 7.3. If the set-valued attribute ***authored-by*** is most often accessed by traversing its value, then we could apply the above schema transformation to refine it into a sequence-valued attribute. The implementation also contains more information, because we could now query the first author of a book, for example.

8 Conclusion

We have developed a formal basis of correct schema transformations. In particular, schemas are formalized as ADTs, schema transformations are formalized as signature interpretations, and correct schema transformations are formalized as information-preserving signature interpretations.

Compared with existing approaches, our formalism captures transformations of all schema components, making it possible to transform uniformly constraints and queries along with structures. In addition, our formalism captures schema transformations between different data models as easily as those within the same data model.

We have compared in detail our notion of information preservation with the most widely used correctness criteria—Hull's notion of relative information capacity. Our notion is strictly less restrictive than calculus dominance, strictly more restrictive than absolute dominance, and incomparable to generic or internal dominance. Moreover, our notion captures more schema transformations that are natural, and fewer schema transformations that are unnatural, than Hull's notion. We have also solved an open problem with Hull's notion—calculus and internal dominance are, respectively, strictly more restrictive than generic and absolute dominance.

Our work lays the foundation of a transformational framework of schema manipulations. Popular transformations of common ADTs can be encoded as transformation rules and proven correct once. Schemas specified with common ADTs can be simplified, restructured, and translated by applying these rules repeatedly. We can also apply these rules to refine schema specifications into efficient storage representations. As examples, we have shown the correctness of common schema transformations that have been proposed in the literature.

Acknowledgment

The author thanks Richard Hull for valuable discussions and comments, which helped improve the presentation of this paper.

References

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

[2] Y. Bar-Hillel and R. Carnap. An outline of a theory of semantic information. In Y. Bar-Hillel, editor, *Language and Information*, chapter 15, pages 221–274. Addison-Wesley, 1964.

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

[4] C. Beeri. Theoretical foundations for OODB's — a personal perspective. *IEEE Data Engineering Bulletin*, 14(2):8–12, June 1991.

[5] C. Beeri. New data models and languages — the challenge. In *Proceedings of the Eleventh ACM Symposium on Principles of Database Systems*, pages 1–15, 1992.

[6] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *Proceedings of the Third International Conference on Extending Database Technology*, 1992.

[7] M. A. Casanova, L. Tucherman, A. L. Furtado, and A. P. Braga. Optimization of relational schemas containing inclusion dependencies. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, pages 317–325, 1989.

[8] M. A. Casanova and V. M. P. Vidal. Towards a sound view integration methodology. In *Proceedings of the Second ACM Symposium on Principles of Database Systems*, pages 36–47, 1983.

[9] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[10] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.

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

[12] R. Hull and C. K. Yap. The format model: A theory of database organization. *Journal of the ACM*, 31(3):518–537, July 1984.

[13] A. Klug. Calculating constraints on relational expressions. *ACM Transactions on Database Systems*, 5(3):260–290, September 1980.

[14] V. M. Markowitz and A. Shoshani. Representing extended entity-relationship structures in relational databases: A modular approach. *ACM Transactions on Database Systems*, 17(3):423–464, September 1992.

[15] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Schema equivalence in heterogeneous systems: Bridging theory and practice. *Information Systems*, 19(1):3–32, January 1994.

[16] J.-M. Nicolas and H. Gallaire. Data base: Theory vs. interpretation. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 33–54. Plenum Press, 1978.

[17] A. Rosenthal and D. Reiner. Tools and transformations—rigorous and otherwise—for practical database design. *ACM Transactions on Database Systems*, 19(2):167–211, June 1994.

[18] S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, April 1994.

[19] D. Tsichritzis and F. Lochovsky. *Data Models*. Prentice-Hall, 1982.

[20] J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 1. Computer Science Press, 1988.

[21] V. M. P. Vidal and M. Winslett. Preserving update semantics in schema integration. In *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 263–271, 1994.

[22] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Vol. B: Formal Models and Semantics*, chapter 13, pages 675–788. MIT Press/Elsevier, 1990.