Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice (Extended Abstract)*

R. J. Miller ** Y. E. Ioannidis *** R. Ramakrishnan †

Department of Computer Sciences, University of Wisconsin-Madison
{rmiller, yannis, raghu}@cs.wisc.edu

1 Introduction

Current theoretical work offers measures of schema equivalence based on the information capacity of schemas. This work is based on the existence of abstract functions satisfying various restrictions between the sets of all instances of two schemas. In considering schemas that arise in practice, however, it is not clear how to reason about the existence of such abstract functions. Further, these notions of equivalence tend to be too liberal in that schemas are often considered equivalent when a practitioner would consider them to be different. As a result, practical integration methodologies have not utilized this theoretical foundation and most of them have relied on ad-hoc approaches.

We present results that seek to bridge this gap. First, we consider the problem of deciding information capacity equivalence and dominance of schemas that occur in practice, i.e., those that can express inheritance and simple integrity constraints. We show that this problem is undecidable. This undecidability suggests that in addition to the overly liberal nature of information capacity equivalence, we should look for alternative, more restrictive notions of equivalence that can be effectively tested. To this end, we develop several tests that each serve as sufficient conditions for information capacity equivalence or dominance. Each test is characterized by a set of schema transformations in the following sense: a test declares that Schema S1 is dominated by Schema S2 if and only if there is a sequence of transformations that converts S1 to S2. Thus, each test can be understood essentially by understanding the individual transformations used to characterize it. Each of the transformations we consider is a local, structural schema change with a clear underlying intuition. These tests permit reasoning about the equivalence and dominance of quite complex schemas. Because our work is based on structural transformations, the same characterizations that underly our tests can be used to guide designers in modifying a schema to meet their equivalence or dominance goals.

* In Extending Database Technology (EDBT), Cambridge, U.K., March 1994.
** Partially supported by NSF Grant IRI-9157368.
*** Partially supported by NSF Grants IRI-9113736 and IRI-9157368 (PYI Award) and by grants from DEC, IBM, HP, and AT&T.
† Partially supported by a David & Lucile Packard Fellowship, by NSF PYI Award and NSF grant IRI-9011563, and by grants from DEC, Tandem, and Xerox.

2 Motivation

Schema equivalence plays a central role in many schema integration tasks. For example, algorithms for detecting equivalent schemas can be used to automate the detection and resolution of structural schema mismatches (or type conflicts). Schema equivalence also plays an important and less recognized role in many other problems encountered in heterogeneous systems. Below, we describe one such problem, that of providing automated support for ad hoc changes to a schema that is being used as a view onto data stored under another schema.

Consider a schema translation tool in which a schema is translated into a schema in a different data model. Many tools produce a translated schema that can be used as a view to pose queries on data stored under the original schema. Within such tools, the translation process produces not only the translated schema, but a set of correspondences between the schemas that defines how an instance of the former corresponds to an instance of the latter. We call these correspondences *instance mappings*. For example, the Pegasus import tool [2] translates relational schemas to Iris schemas (Iris is a functional object model). For each Iris type, the result of translation includes a rule over a collection of relations in the original schema that defines the instances of the type.

Such translation tools fully automate the production of instance mappings. A designer need only be concerned with the resulting schema; all details of establishing schema correspondences are hidden. We now want to permit the designer to change the translated schema. Again, we want the tool to automatically infer and record any changes necessary to the instance mapping.

For example, suppose Schema R1 of Figure 1 is produced by a translation tool from an underlying schema in another data model. A designer may wish to change the default translation and represent `Grant` as an attribute of *Workstation* not *Project* as in Schema R2. If the tool can test for equivalence (or dominance) and automatically produce an instance mapping between schemas, then the designer does not need to manually update the instance mapping as a result of this change. Currently, translation tools, such as Pegasus, do not give such support for ad hoc view changes. Rather, they provide some form of data definition language in which default mappings are expressed and which may be used by a designer to manually change a mapping.

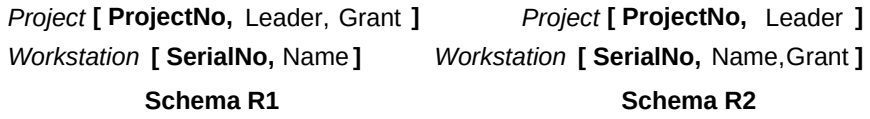

Fig. 1. Parts of two relational schemas. Keys are depicted in bold.

This problem is clearly not restricted to translation and applies to a number of applications in heterogeneous databases in which one schema is maintained

as a view over other schemas. Our study of schema equivalence has been motivated by the needs of such applications. For these applications, the notion of equivalence must be based on the capacity of schemas to store information. In addition to algorithms for producing equivalent schemas (this is the translation or transformation problem), these applications also require algorithms for both deciding if two schemas are equivalent and for producing the correspondence between the schemas (that is, an instance mapping).

3 Schema Intension Graphs

In this section, we briefly sketch the basic constructs of the *Schema Intension Graph* (SIG) data model, which will be used to present our results. A discussion of the motivation for using SIGs and a full definition are given elsewhere [8].

The basic building blocks of the model are sets of data values (represented by the nodes of a graph). These sets may be combined by nested applications of union and product constructors to form new sets. The model also permits the expression of binary relations between pairs of sets (represented by graph edges) and simple constraints on them, i.e., totality, surjectivity, functionality and injectivity (represented by annotations on the edges).

Let $\mathcal{T}$ be a finite set of mutually exclusive abstract types, where each $\tau \in \mathcal{T}$ is an infinite set of symbols. The universe U is the union of symbols in all types. Let $\mathcal{T}^*$ be the closure of $\mathcal{T}$ under finite products and sums. A SIG is a graph, $G = (N, E)$, defined by two finite sets N and E. The set N contains *simple nodes* and *constructed nodes*, which are the products and sums of other nodes. Each simple node $A \in N$, is assigned a type, $\tau(A) \in \mathcal{T}^*$. The type of a constructed node is the product or union of the types of its constituent nodes. The set E contains labeled *edges* between nodes in N. Each $e \in E$ is denoted $e : A - B$, for $A, B \in N$. For each edge $e \in E$, its inverse, denoted e°, is in E. If $\tau(A) = \tau(B)$ then $e : A - B$ may optionally be designated as a selection edge. We use the term constraint to refer to any annotation or selection constraint on an edge.

An *instance* $\Im$ of G is a function whose domain is the sets N of nodes and E of edges. For each simple node, $A \in N$, $\Im[A]$ is a finite subset of $\tau(A)$. For each product node, $A \times B \in N$, $\Im[A \times B]$ is the full cross product of the sets $\Im[A]$ and $\Im[B]$. For each sum node, $A + B \in N$, $\Im[A + B]$ is the union of the sets $\Im[A]$ and $\Im[B]$. For each edge, $e : A - B \in E$, $\Im[e]$ is any subset of the product of $\Im[A]$ and $\Im[B]$. For each selection edge, $\sigma : A - B$, $\Im[\sigma]$ is a subset of the identity relation on $\Im[A]$. The set of all instances of G is denoted $I(G)$.

An annotation of a SIG $G = (N, E)$ is a function $\mathcal{A}$ whose domain is the set of edges E. For all $e \in E$, $\mathcal{A}(e) \subseteq \{f, i, s, t\}$. A *SIG schema* S is a pair $S = (G, \mathcal{A})$. An instance $\Im$ of G is a *valid instance* of $\mathcal{A}$ if for all $e \in E$, whenever $f \in \mathcal{A}(e)$ (respectively i, s or $t \in \mathcal{A}(e)$), $\Im[e]$ is a functional (respectively injective, surjective or total) binary relation. The set of all valid instances of S is denoted $I(S)$. The set of symbols of an instance, denoted $Sym(\Im)$, is the set of elements of U that appear in the range of $\Im$. For a subset of the universe, $Y \subseteq U$, $I_Y(S)$ denotes the set of instances of S that contain only symbols in Y.

4 Information Capacity

We consider formal notions of correctness for schema transformations that are based on the preservation of the information content of schemas[3, 4, 7, 9]. For a schema S, the latter is the set of valid instances, $I(S)$. Intuitively, a schema $S2$ has more information capacity than a schema $S1$ if every instance of $S1$ can be mapped to an instance of $S2$ without loss of information. Specifically, it must be possible to recover the original instance from its image under the mapping.

Absolute equivalence characterizes the minimum that is required to achieve information capacity equivalence and provides a foundation on which more specialized definitions of equivalence may be built. It is based on the existence of invertible (i.e., injective) maps between the sets of instances of schemas.

Definition 1. An *information (capacity) preserving mapping* between the instances of two schemas $S1$ and $S2$ is a total, injective function $f : I_Y(S1) \rightarrow I_Y(S2)$, for some $Y \subseteq U$. An *equivalence preserving mapping* is a bijection $f : I_Y(S1) \rightarrow I_Y(S2)$.

Definition 2. The schema $S2$ *dominates* $S1$ *absolutely*, denoted $S1 \preceq_{abs} S2$, if there is a finite $Z \subseteq U$ such that for each $Y \supseteq Z$ there exists an information preserving mapping $f : I_Y(S1) \rightarrow I_Y(S2)$. Also, $S1$ and $S2$ are *absolutely equivalent*, denoted $S1 \sim_{abs} S2$, if for each $Y \supseteq Z$ there exists an equivalence preserving mapping $f : I_Y(S1) \rightarrow I_Y(S2)$.

Decidable characterization of absolute equivalence are known for relational schemas with (primary) key dependencies and for types formed by the recursive application of product, set or union constructors on finite and infinite base types [1, 4, 5]. SIGs permit the representation of sets formed from nested product and union constructors, as well as simple constraints between these sets. These additions make testing for equivalence (and therefore dominance undecidable.

Theorem 3. *Testing for absolute equivalence of SIGs is undecidable.*

In principle, arbitrary mappings f may be used to satisfy the definitions of absolute dominance and equivalence. In fact, the definitions do not even require that the mappings can be finitely specified; they can simply be an infinite list of pairs of schema instances. Clearly, such mappings are of little use in a practical system. Furthermore, there exist very simple schemas with no "natural" correspondence between them that satisfy the definition of absolute dominance through a very complex instance level mapping [4]. This result, coupled with our undecidability result, show that absolute equivalence and dominance do not provide a sufficient foundation for analyzing practical integration problems.

To overcome the limitations of absolute equivalence, various abstract properties have been proposed that restrict the class of allowable instance mappings [4]. For example, *internal equivalence* states that two instances can be associated by an instance mapping only if they contain (almost) the same set of symbols. However, testing for both internal equivalence and dominance of SIG schemas is also undecidable [8].

5 Structural Transformations

Given Theorem 3 and other similar results, the question remains as to how practitioners can develop rigorous methodologies. Our response is to propose sets of schema transformations on SIGs that preserve or augment information capacity and are similar to transformations in existing integration and translation methodologies [3, 6, 7, 9]. For each set of transformations, we characterize precisely when a schema can be created from another through any arbitrary sequence of transformations. Our characterizations are couched in terms of definitions of dominance and equivalence having the following properties: 1-each is a sufficient condition for internal dominance or equivalence, respectively; 2-each is complete for a given set of transformations; and 3-each leads to a decidable procedure for testing if one schema can be transformed into another and for producing an information preserving (respectively, equivalence preserving) instance mapping between the schemas.

In the following definitions, $S1$ and $S2$ denote SIG schemas. A transformation T on $S1$ that produces $S2$ is denoted by $S1 \xrightarrow{T} S2$. An arbitrary (possibly empty) sequence of transformations $\xrightarrow{T}$ is denoted $\xrightarrow{T}^{*}$. For each class of transformations, an intuitive description is presented first, followed by a formal definition.

5.1 Definition of the Transformations

An *annotation transformation* (or α-transformation) removes constraints from an edge of a SIG schema. An example is shown in Figure 2.

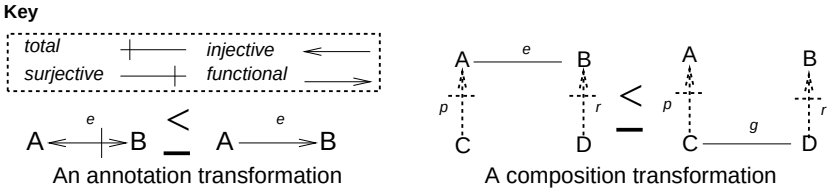

Fig. 2. An α-transformation and a $\circ$-transformation.

Definition 4. Let $S1$ contain an edge e. Let $S2$ be identical to $S1$ except that the constraints on e in $S2$ may be any subset of the constraints on e in $S1$. Then $S1 \xrightarrow{\alpha_e} S2$ and α_e is called an *annotation transformation (α-transformation).*

Theorem 5. *If* $S1 \xrightarrow{\alpha} S2$ *then* $S1 \preceq_{int} S2$.[5]

[5] Subscripts or superscripts indicating specific edges or nodes involved in the transformations may be omitted in denoting a transformation.

A *composition transformation* (or $\circ$-transformation) replaces an edge $e : A - B$ with another edge $g : C - D$. An example is shown in Figure 2. Such transformations permit attributes and entities to be moved in a schema. The instance mapping populates an instance of the edge g with an instance of the path $r^{\circ} \circ e \circ p$.

Definition 6. Let $e : A - B$ be an edge of $S1$ and let $p : C - A$ and $r : D - B$ be (possibly trivial) surjective functional paths in $S1$ not containing e. Let $G2 = G1$ except e is replaced by $g : C - D$ and the constraints on g in $S2$ are exactly the constraints on the path $r^{\circ} \circ e \circ p$ in $S1$. Then $S1 \xrightarrow{\circ_g^e} S2$ is called a *simple composition transformation* (a *simple* $\circ$*-transformation*).

Theorem 7. *Let $\circ_g^e$ be a simple $\circ$-transformation that uses the surjective functional paths p and r. If $S1 \xrightarrow{\circ_g^e} S2$ then $S1 \preceq_{int} S2$. If $\mathcal{A}1(p) = \mathcal{A}1(r) = \{f, i, s, t\}$ and constraints of g are equal to the constraints of e then $S1 \sim_{int} S2$.*

We can also construct information preserving mappings for simple $\circ$-transformations applied in parallel and $\circ$-transformations that move an edge to multiple edges. We therefore define a larger class of transformations.

Definition 8. A $\circ$*-transformation* is a set of one or more simple $\circ$-transformations. A $\circ$-transformation is denoted $S1 \xrightarrow{\{\circ_{g1}^{e1}, \ldots \circ_{gn}^{en}\}} S2$ where the $\circ_{gi}^{ei}$ are simple $\circ$-transformations and all g_i are distinct.

Theorem 9. *If $S1 \xrightarrow{\{\circ_{g1}^{e1}, \ldots \circ_{gn}^{en}\}} S2$ then $S1 \preceq_{int} S2$. If each component simple $\circ$-transformation is equivalence preserving and all e_i are distinct then $S1 \sim_{int} S2$.*

A *selection transformation* (or ς-transformation) creates or deletes nodes and edges. The ς-transformations we consider are depicted in Figure 3.

Fig. 3. Selection transformations.

A *node creation* ς*-transformation* creates a new node that is isomorphic to an existing node. A bijective selection edge between the two nodes enforces the constraint that the nodes be assigned identical sets in any valid instance.

Definition 10. Let A be a node of $S1$. Let A' be a new node not in $S1$ and $\sigma_{A'} : A \leftarrow\!\!\!\!\!\!\rightarrow A'$ a new bijective selection edge. Let $S2$ be $S1$ with the addition of A' and $\sigma_{A'}$. Then, $S1 \xrightarrow{\varsigma_A} S2$ is called a *node creation* ς*-transformation*, and $S2 \xrightarrow{\varsigma_A} S1$ is called a *node deletion* ς*-transformation*.

An *edge creation* ς*-transformation* creates a new edge. To preserve information capacity, the new edge is a bijective selection edge on a node.

An *edge deletion* ς*-transformation* removes an edge. If information capacity is to be preserved, arbitrary edges cannot be removed from a SIG. However, instances of bijective selection edges are fully defined by the instances of the incident nodes. Such edges may therefore be removed.

Definition 11. Let $\sigma_A : A \leftarrow\!\!\!\!\!\!\rightarrow A'$ be a new bijective selection edge between nodes A and A' of $S1$ and let $S2$ be $S1$ with the addition of σ_A. Then, $S2 \xrightarrow{\varsigma_{\sigma_A}} S1$ is an *edge deletion* ς-transformation and if $A = A'$ then $S1 \xrightarrow{\varsigma_{\sigma_A}} S2$ is an *edge creation* ς-transformation.

Theorem 12. *Let* $S1 \xrightarrow{\varsigma} S2$*. If* ς *is a node creation, node deletion or edge creation* ς*-transformation, then* $S1 \sim_{int} S2$*. If* ς *is an edge deletion* ς*-transformation that removes an edge from a node to itself, then* $S1 \sim_{int} S2$*, otherwise,* $S1 \preceq_{int} S2$*.*

When ς-transformations, $\circ$-transformations and α-transformations are combined, they permit complex additions and modifications to be made to a schema. Also, the information preserving mappings created by ς-transformations, $\circ$-transformations and α-transformations can be composed.

Corollary 13. *If* $S1 \xrightarrow{\alpha \circ \varsigma}{}^{*} S2$ *then* $S1 \preceq_{int} S2$*.*

5.2 Characterization of Dominance and Equivalence

We develop a characterization of dominance and equivalence that is complete with respect to all three types of transformations considered.

Definition 14. $S2$ $\alpha\circ\varsigma$-dominates $S1$, denoted $S1 \preceq_{\alpha\circ\varsigma} S2$, if there exist a surjective, injective node map $\psi : N1 - N2$ and a surjective, injective edge map $\theta : (E1 \cup N1) - E2$ satisfying the following.

1. If $A \in N1$ and ψ is not defined on A, then there exists a bijective selection path in $S1$ from A to a node B where ψ is defined on B.
2. If $A \in N1$ then for all $g' \in \theta(A)$ (where $g' \in E2$ and $g' : C' - D'$, $\psi^{-1}(C') = C$ and $\psi^{-1}(D') = D$), there exist surjective functional paths $p : C - A$ and $r : D - A$ in $S1$ and the constraints on g' in $S2$ are a subset of the constraints on the path $r^{\circ} \circ p$ in $S1$.
3. If $e \in E1$ and θ is not defined on e, then e is a bijective selection edge in $S1$.
4. If $e : A - B \in E1$ then for all $g' \in \theta(e)$ (where $g' \in E2$ and $g' : C' - D'$, $\psi^{-1}(C') = C$ and $\psi^{-1}(D') = D$), there exist surjective functional paths $p : C - A$ and $r : D - B$ in $S1$ (not containing e) and the constraints on g' in $S2$ are a subset of the constraints on the path $r^{\circ} \circ e \circ p$ in $S1$.

$S1$ is $\alpha\circ\varsigma$*-equivalent* to $S2$, denoted $S1 \sim_{\alpha\circ\varsigma} S2$, if $S1 \preceq_{\alpha\circ\varsigma} S2$ and $S2 \preceq_{\alpha\circ\varsigma} S1$.

One can show that $\alpha \circ \varsigma$-dominance is complete with respect to ς-transformations, $\circ$-transformations and α-transformations. This result and Corollary 13 imply that $\alpha \circ \varsigma$-dominance is a sufficient condition for internal dominance.

Theorem 15. *Let $S1$ and $S2$ be two SIG schemas. Then, $S1 \xrightarrow{\alpha \circ \varsigma}{}^{*} S2$ iff $S1 \preceq_{\alpha \circ \varsigma} S2'$ where $S2' \cong S2$.*

5.3 Testing for Dominance

Definition 14 essentially gives an algorithm to test if two SIG schemas are in an $\alpha \circ \varsigma$-dominance relation, which by Theorem 15, also determines if a schema may be obtained from another through a sequence of transformations. Furthermore, our proof is constructive so we have the following result.

Corollary 16. *If $S1 \preceq_{\alpha \circ \varsigma} S2$ then we can construct a sequence of α-transformations, $\circ$-transformations and ς-transformations such that $S1 \xrightarrow{\alpha \circ \varsigma}{}^{*} S2$ via this sequence and an information preserving mapping $f : I(S1) \rightarrow I(S2)$.*

The complexity of algorithms for testing $\alpha \circ \varsigma$-equivalence and $\alpha \circ \varsigma$-dominance and producing instance mappings is examined elsewhere [8].

References

1. S. Abiteboul and R. Hull. Restructuring Hierarchical Database Objects. *Theoretical Computer Science*, 62:3–38, 1988.
2. J. Albert, R. Ahmed, M. A. Ketabchi, W. Kent, and M. C. Shan. Automatic Importation of Relational Schemas in Pegasus. In *Proc. of the 3rd Int'l Workshop on Research Issues in Data Eng.: Interoperability in Multidatabase Systems*, pages 105–113, Vienna, Austria, Apr. 1993.
3. C. F. Eick. A Methodology for the Design and Transformation of Conceptual Schemas. In *Proc. of the Int'l Conf. on Very Large Data Bases*, pages 25–34, Barcelona, Spain, Sept. 1991.
4. R. Hull. Relative Information Capacity of Simple Relational Database Schemata. *SIAM Journal of Computing*, 15(3):856–886, Aug. 1986.
5. R. Hull and C. K. Yap. The Format Model: A Theory of Database Organization. *Journal of the ACM*, 31(3):518–537, 1984.
6. L. A. Kalinichenko. Methods and Tools for Equivalent Data Model Mapping Construction. In *Proc. of the Int'l Conf. on Extending Database Technology*, pages 92–119, Venice, Italy, Mar. 1990.
7. 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, Sept. 1992.
8. R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice. *Information Systems*, 19(1):3–31, 1994.
9. A. Rosenthal and D. Reiner. Theoretically Sound Transformations for Practical Database Design. In *Proc. of the Int'l Conf. on Entity-Relationship Approach*, pages 115–131, New York, NY, Nov. 1987.

This article was processed using the LaTeX macro package with LLNCS style