Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice [*]

R. J. Miller[†] **Y. E. Ioannidis**[‡] **R. Ramakrishnan**[§]

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

December 1, 1993

Abstract

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. We demonstrate the power of these tests by showing that one can reason 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.

[*]This paper appears in: *Information Systems*, 19(1):3-31, 1994. An extended abstract appears in: *Extending Database Technology (EDBT)*, Cambridge, U.K., March 1994.

[†]R. J. Miller has been partially supported by the National Science Foundation (NSF) under Grant IRI-9157368.

[‡]Y. Ioannidis has been partially supported by the NSF under Grants IRI-9113736, IRI-9224741, and IRI-9157368 (PYI Award), and by grants from DEC, IBM, HP, AT&T, and Informix.

[§]R. Ramakrishnan has been partially supported by a David and Lucile Packard Foundation Fellowship in Science and Engineering, by the NSF under a PYI Award and under grant IRI-9011563, and by grants from DEC, Tandem, and Xerox.

1 Introduction

The problem of schema translation is to transform an existing schema in a given data model into an equivalent schema, possibly in a different data model. A closely related problem is that of schema integration. At the heart of schema integration lies the problem of detecting if two schemas or parts of schemas are equivalent. Theoretical work on schema integration and translation has focused on the development of notions of equivalence for schemas. Such work, while mathematically elegant, does not address many practical issues and is not an adequate foundation for developing pragmatic solutions. On the other hand, without a good foundation, practical work has been largely ad hoc. Solutions are motivated by the needs of specific classes of examples and often do not generalize.

In this paper, we present formal results that provide a basis for developing practical schema integration and translation techniques. Our previous work has highlighted the need for a formal notion of equivalence [14]. Specifically, we examined the notion of *relative information capacity* [8] and identified anomalies that can arise when using transformations that do not guarantee that information capacity is preserved. However, there are only limited theoretical results on information capacity equivalence and dominance. In addition, to be directly usable in a practical context, a decidable characterization of both equivalence and dominance of schemas is needed. In this paper, we examine the characterizations that exist for various classes of schemas and extend these results by showing that no characterization is possible even for a more general though still simple class.

Furthermore, while being a required condition, information capacity is not sufficient to guarantee a natural correspondence between schemas. In defining equivalence preserving transformations of schemas, practitioners use their own intuition about what constitutes a valid structural correspondence between schemas. Our undecidability result suggests that in addition to the overly liberal nature of information capacity equivalence, we should look for alternative, more restrictive notions of equivalence and dominance 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. Our earlier work showed that for many practical tasks, it is sufficient to ensure the information capacity dominance of schemas [14]. We therefore consider transformations that preserve the information capacity of a schema and transformations that augment it. Our transformations are presented formally in terms of the *Schema Intension Graph* (SIG) data model, which is defined in Section 3. Informally, we consider transformations that permit common constraints on a schema to be removed. These constraints can express key dependencies, integrity constraints and other common dependencies between sets. Other transformations permit attributes or entities to be moved or copied. As a simple example, we can transform Schema S1 of Figure 1 into Schema S2. The **Phone** entity is copied from **Address** to **Person** and the **Degree** entity is moved. We characterize exactly when such transformations preserve or augment information capacity (that is, when the transformations are valid). Additionally, the transformations define the constraints that must hold on the transformed schema based on the constraints on original schema. Our characterizations let us test exactly when an arbitrary schema can be obtained from another through some sequence of transformations.

We demonstrate the power of these tests by showing that one can reason about the equivalence and dominance of 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 schemas to meet their equivalence or dominance goals. We have defined and used a formal model that permits reasoning about schemas at a level that is easily translated into the terms of most commonly used models. It is not necessary for a designer to understand our SIG data model or the details of our results to be able to benefit from the algorithms we propose.

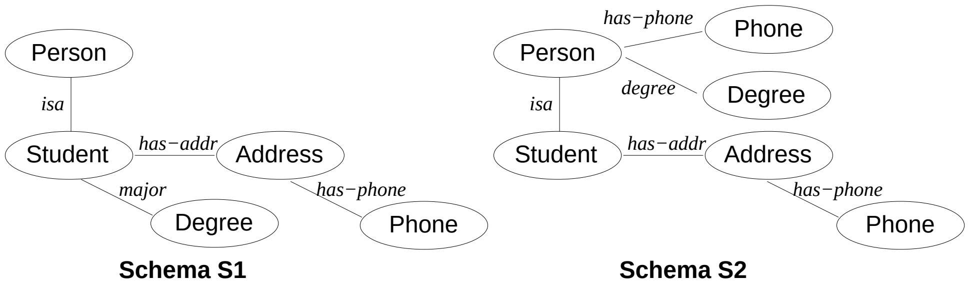


Figure 1: Example transformation of an informal object-oriented schema.

2 Motivation

Before conducting a study of schema equivalence, the question of why this is important must be addressed. Below, we present examples belonging to two broad classes of problems whose solutions depend on having effective procedures to determine if two schemas are equivalent. In addition to showing the importance of schema equivalence in practical problems, these motivating examples provide insight into the type of results on schema equivalence that may have direct practical import. The first class of problems is that of automating the integration of schemas. We focus on one problem in this class, that of detecting and resolving what has been termed structural mismatches or conflicts in schemas that are being integrated. The second class is 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.

2.1 Schema Integration

Algorithms for determining the equivalence and dominance of schemas play a number of important roles in the integration process. For clarity, we limit our discussion here to a specific application of schema equivalence.

Integration methodologies vary a great deal but at a very coarse level they often include the following steps.[1] An initial comparison is performed in which correspondences among schemas are determined. For example, the attribute `ProjectNo` in some schema could be identified as being identical to the attribute `PNo` in another. After general correspondences are known, conflicts in the schemas may be detected and resolved. Conflicts include type conflicts (also called "structural mismatches") in which similar information has been represented using different schema constructs (for example, as a relationship in one schema and as an entity in another) or different groupings of constructs (for example, sets of attributes are grouped differently in relational tables, as in the example below). Type conflicts are typically resolved by changing the constructs used in one or more of the schemas so that all schemas use identical constructs. This step can be viewed as a goal driven restructuring of the original schemas based on information about other schemas to be integrated. After conflict resolution, the schemas are merged. For example, a single attribute `ProjNo` in the integrated view may represent information from the attribute `ProjectNo` of one schema and the attribute `PNo` of another.

The resolution strategy for type conflicts may be a fixed choice of one construct over another or may be left up to the schema designer [4, 20]. However, certain choices are simply wrong in certain situations and may cause errors if the integrated view is used to retrieve or store data in the underlying schemas [14]. If the integration methodology makes use of information about the equivalence of schemas, such wrong choices can be avoided. We illustrate this point using a simple relational example.

Consider the integration of two complex relational schemas, R1 and R2. These schemas contain (among other tables) the *Project* and *Workstation* tables depicted in Figure 2. The integrated schema is to be

[1]These steps are outlined in a 1986 survey of integration methodologies [4] and continue to be used in more recent work [19, 20]. We have only presented them in outline form for brevity.

used as a view for accessing information in both schemas, so we want to ensure that the integrated view can express all the data in the original schemas. Furthermore, we need to ensure that we have a set of correspondences that defines how instances of the original schemas collectively correspond to an instance of the integrated view so that queries on the view may be translated to queries on the original schemas. We call these correspondences *instance mappings.* If such mappings exist, we say that the integrated view *dominates* the original schemas. Formal definitions of dominance and equivalence of schemas are given in Section 4.

Schema R1	**Schema R2**
Project [**ProjectNo,** Leader, Grant]	*Project* [**ProjectNo,** Leader]
Workstation [**SerialNo,** Name]	*Workstation* [**SerialNo,** Name, Grant]

Figure 2: Parts of two relational schemas to be integrated. Keys are depicted in bold.

Suppose that a designer has indicated that the information contained in the two *Workstation* tables corresponds (that is, the tables model the same real world entities), as does the information in the two *Project* tables (and similarly, for all attributes depicted in the figure). In the conflict resolution phase, the problem is to pick a common representation for all the information about *Workstations* and *Projects*. Reasoning at an intuitive level, if we can determine a unique `Grant` to associate with each *Workstation* (and if every `Grant` is associated with some *Workstation*), then we can represent any instance of R1 as an instance of R2. The latter condition is needed to ensure that all `Grant` data may be accessed through the view.

Suppose schema R1 also contains the table *Owner*[**SerialNo,** `ProjectNo`] and the proper inclusion dependencies to ensure that every `Grant` is associated with some *Workstation* (*Owner*. **SerialNo** $\subseteq$ *Workstation*.**SerialNo** and *Project*.**ProjectNo** $\subseteq$ *Owner*.`ProjectNo`). We can then correctly represent every instance of R1 as an instance of R2, so we could choose the tables of R2 as the common representation for *Workstation* and *Project* information in the integrated view.

This was just one example of how R2 might have more capacity to store information. It may also be the case that for different schemas the *Workstation* and *Project* tables of R1 dominate R2. The reasoning needed to make such a determination can be quite complex. It depends on the constraints expressed on the schemas (including whether null values are allowed). A designer cannot be expected to intuit a correct representation. Furthermore, a fixed choice of one representation (perhaps always choosing the table with the most attributes) will not be correct for all schemas. If a tool uses algorithms for detecting the equivalence or dominance of schemas, then it can correctly resolve schema conflicts or guide a designer in modifying the schemas using knowledge about the existence of instance mappings.

2.2 Ad Hoc Schema Changes

Algorithms for deciding schema equivalence also have important applications in areas outside of schema integration. We discuss one important class of problems, that of supporting 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 an equivalent schema (typically in a different data model), which 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 an instance mapping between the schemas. 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 2 is produced by a translation tool from an underlying schema in another data model.[2] 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 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.

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 schema(s) [18, 21, 22].

2.3 Discussion

Several requirements on our study of schema equivalence can be identified from these motivating examples. First, for the applications we are considering, the notion of equivalence must be based on the ability or capacity of schemas to store information. Second, practical problems require not only procedures for producing equivalent schemas (this is the translation or transformation problem) but also for testing if two schemas are equivalent. Finally, for equivalence tests to be usable, they must not only produce a decision about the equivalence of schemas, but they must also produce the correspondence between the schemas (that is, an instance mapping).

In the next section, we present the *Schema Intension Graph* (SIG) data model that will be used to present our results. In Section 4, we define information capacity and examine existing results on information capacity equivalence and dominance. We extend these results by considering the information capacity of SIGs and show that it is not always possible to test for the equivalence or dominance of schemas. In Section 5, we provide decision procedures that will allow us to decide equivalence for a large class of schemas. In the examples of Sections 5 and 6, we briefly demonstrate how our results may be used in a practical setting to meet the requirements of common integration and translation problems such as the ones we have just examined.

3 A Formal Data Model

In this section, we define the data model used to present our results. We have chosen to define a new data model for this purpose for two main reasons. First, we need a model that allows us to compare schemas with respect to their information capacity. As we saw in the previous section, given two schemas, a typical question of interest is whether each instance of the first schema can be represented as an instance of the second schema (in such a way that it is possible to 'go back' to the first instance). In order to address questions such as the one above, we need a way to reason about possible schema instances given a set of constraints over them. Other formal studies of information capacity have typically used the relational model [8] or models based on complex types [1, 9, 10, 15]. Instead of extending complex types with constraints, we chose to use a model in which constraints are expressed on collections of entities of an instance rather than on the internal structure of a single entity. We explore this point further after we have presented our model. Second, to simplify our task, we wish to include in the model only a minimum set of constructs and constraints necessary to model a large class of commonly occurring schemas. Furthermore, we require that any reasoning about schema equivalence be done in a form that is easily conveyed back to a schema designer. To aid in this goal, we strive to meet requirements laid out by practitioners in this field [17]. Specifically, the constraints we include in the model are local (that is, they are robust to schemas changes), comprehensible (easily understood and used by a database designer), and not based on unrealistic assumptions about the set of valid instances of a schema. They also appear in some form in most common data models, and are therefore widely recognized as being useful.

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 and simple integrity constraints

[2]In reality, most tools translate into more expressive data models but, for brevity, we reuse the same relational example.

(dependencies) on these binary relations. The binary relations are represented by edges of a graph and the constraints by annotations on the edges. The constraints include totality and surjectivity, which express that every element of the first or the second set must participate in an instance of the binary relation, respectively, and functionality and injectivity, which express that an element of the first or the second set may appear at most once in an instance of the binary relation, respectively. [3]

3.1 Schema Intension Graphs

Let Λ be an infinite set of symbols that will serve as labels for schema constructs. Let $\mathcal{T}$ be an infinite set of simple abstract types. Let $\mathcal{T}^*$ be the closure of $\mathcal{T}$ under finite products and sums. Each simple type $\tau \in \mathcal{T}$ is an infinite set of symbols. All simple types are pairwise disjoint and disjoint from the set of labels Λ. The universe U is the union of symbols in all types of $\mathcal{T}$.

A *schema intension graph* (SIG) is a graph, $G = (N, E)$, defined by two finite sets N and E. The set N contains a set of symbols $M \subseteq \Lambda$. The nodes in M are called *simple nodes*. Additionally, N may contain *constructed nodes* that are the products and sums of other nodes, where

- if $A, B \in N$ then the node $A \times B$ may be in N; and
- if $A, B \in N$ then the node $A + B$ may be in N.

Each simple node $A \in N$, is assigned a type, $\tau(A) \in \mathcal{T}^*$. The type of a constructed node is the cross-product or union of the types of its constituent nodes. Multiple nodes may have the same type.

Each element $e \in E$ is a labeled edge between two nodes of N. An edge e is denoted $e : A - B$, indicating it is an edge from node A to node B. For each edge $e \in E$, the inverse of e, denoted e°, is in E. The set E may contain arbitrary edges between nodes as well as multiple edges between the same pair of nodes. If $\tau(A) = \tau(B)$, then an edge $e : A - B$ may optionally be designated as a selection edge and is denoted by a label σ_A^B. If $\tau(A) = \tau(B) \times \tau(C)$ for some node C, then $e : A - B$ may optionally be designated as a projection edge and is denoted by a label Π_A^B. (When no confusion can arise subscripts and/or superscripts on projection and selection edges will be omitted.) An instance of a SIG is constrained to give all selection and projection edges special interpretations.

3.1.1 Instances

In a SIG, the nodes represent typed domains and the edges represent abstract morphisms between domains. An instance of a graph is an assignment of a specific set of elements of the appropriate type to each node and specific binary relations to the edges of a graph. An instance corresponds to a specific database state.

An *instance* of G is a function whose domain is the sets N of nodes and E of edges. The set of all instances of G is denoted $I(G)$. An instance $\Im \in I(G)$ is restricted as follows.

- For each simple node $A \in N$, $\Im[A]$ is a finite set of elements where $\Im[A] \subseteq \tau(A)$.
- For each product node $(A \times B) \in N$, $\Im[A \times B]$ is the cross product of elements from $\Im[A]$ and $\Im[B]$, $\Im[A \times B] = \Im[A]\ \mathrm{X}\ \Im[B]$. (Here X denotes ordinary cartesian product of sets.)
- For each sum node $(A + B) \in N$, $\Im[A + B]$ is the union of elements from $\Im[A]$ and $\Im[B]$, $\Im[A + B] = \Im[A] \cup \Im[B]$.
- For each edge $e : A - B \in E$, $\Im[e]$ is a subset of the cross product of elements from $\Im[A]$ and $\Im[B]$, $\Im[e] \subseteq \Im[A]\ \mathrm{X}\ \Im[B]$. For the edge e°, $(b, a) \in \Im[e^\circ]$ iff $(a, b) \in \Im[e]$.
- For each selection edge $\sigma_A^B : A - B$, $\Im[\sigma_A^B]$ is a subset of the identity relation on $\Im[A]$.
- For each projection edge $\Pi_A^B : A - B$, (where $\tau(A) = \tau(B) \times \tau(C)$ for some C), $\Im[\Pi_A^B]$ is the projection of $\Im[B]$ components from $\Im[A]$. Namely, $\Im[\Pi_A^B] = \{((b, c), b) \mid (b, c) \in \Im[A] \text{ and } b \in \Im[B]\}$.

[3]For those familiar with category theory, SIG schemas form a simple class of categories where the nodes are finite sets and the arrows are binary relations on pairs of sets [3].

3.1.2 Annotations

Each edge of a SIG is annotated with a (possibly empty) set of properties. Each property is a constraint that restricts the set of valid binary relations that may be assigned to an edge by an instance. An instance of a SIG is a *valid instance* of a set of annotations if the binary relation assigned to each edge satisfies all annotations on the edge.

The following four properties are used to annotate edges of SIGs: totality, surjectivity, functionality and injectivity. Let A and B be two sets. A binary relation $r : A - B$ is *total* (denoted $e : A {+\!\!-} B$) if it is defined for all elements of A; *surjective* ($e : A {-\!\!+} B$) if it is defined for all elements of B; *functional* ($e : A \longrightarrow B$) if an element of A determines at most one element of B; and *injective* ($e : A \longleftarrow B$) if an element of B determines at most one element of A. Also, a *bijection* is a total, surjective, injective function. Note that all four properties are independent in that no subset of the properties expressed on a relation between arbitrary sets A and B logically implies any property not in that subset.

An annotation of a SIG $G = (N, E)$ is a function $\mathcal{A}$ whose domain is the set of edges E. For all $e \in E$, $\mathcal{A}(e) \subseteq \{f, i, s, t\}$. An instance $\Im$ of G is a *valid instance* (also called a *model*) of $\mathcal{A}$, denoted $\Im \models \mathcal{A}$, if for all $e \in E$, if $f \in \mathcal{A}(e)$ (respectively i, s or $t \in \mathcal{A}(e)$) then $\Im[e]$ is a functional (respectively injective, surjective or total) binary relation. A *SIG schema* S is a pair $S = (G, \mathcal{A})$. In what follows, when discussing a SIG schema Si, it will be assumed that $Si = (Gi, \mathcal{A}i)$ and $Gi = (Ni, Ei)$.

The set of instances of S is the set of all instances of G that model $\mathcal{A}$. That is, $I(S) = \{\Im \mid \Im \in I(G)$ and $\Im \models \mathcal{A}\}$. 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, $I_Y(S) = \{\Im \mid \Im \in I(S)$ and $Sym(\Im) \subseteq Y\}$.

The restriction of an edge to being a selection or projection edge can also be viewed as a constraint on the set of valid instances of an edge. We use the term constraint to refer to any annotation, selection constraint or projection constraint.

Edges, like binary relations, can be composed. If $r : A - B$ is a binary relation then for $a \in A$, $r(a)$ denotes the set of elements of B associated with the element a in r. Additionally, if $s : B - C$ is a binary relation then $s \circ r : A - C$ denotes the composition where $s \circ r(a) = s(r(a))$. Similarly, for compositions of edges, $e_2 \circ e_1$ means e_1 followed by e_2. The composition of two functional (respectively injective, surjective or total) binary relations is also functional (respectively injective, surjective or total). This motivates the following definition.

Definition 3.1 Let $G = (N, E)$ be a SIG and $\mathcal{A}$ an annotation function on G. A *path*, $p : N_1 - N_k$, in G is a (possibly empty) sequence of edges $e_1 : N_1 - N_2$, $e_2 : N_2 - N_3$, ..., $e_{k-1} : N_{k-1} - N_k$ and is denoted $e_{k-1} \circ e_{k-2} \circ ... \circ e_1$. A path is called functional (respectively injective, surjective or total) if every edge in the path is functional (respectively injective, surjective or total). Similarly, a path is called a selection path if every edge on the path is a selection. A path is called a projection path if every edge on the path is a projection or selection and at least one edge is a projection. The trivial path is a path from a node to itself containing no edges. The trivial path satisfies all constraints.[4] •

Paths are denoted by dashed lines in figures.

3.2 An Example SIG

Figure 3 depicts an example SIG schema. Selection edges can be used to model both specialization and generalization. The nodes **Student** and **Professor** are subsets of **Employee**. There may be elements of **Employee** that are not in **Student** or **Professor**, so **Student** and **Professor** are both specializations of **Employee**. However, the node **Student** is the generalization (that is, the exact sum) of the nodes **TA** and **RA**. The bijective selection edge between **Student** and **TA** + **RA** enforces the constraint that every **Student** is either an **RA** or a **TA**. The edge *teaches* represents the fact that every course is taught by a single professor. Furthermore, each course has a single title and text book as represented by the *attr* edge.

[4]Alternatively, identity edges from each node to itself may have been included in the definition of SIGs. This latter choice is more consistent with the view of SIGs as categories.

Additionally, selection edges from the **TA + RA** node indicate that the set of all TAs and the set of all RAs may be selected out of the constructed node. Similarly, the projection edges from **Text** × **Title** contain the projection of **Text** and **Title** values.

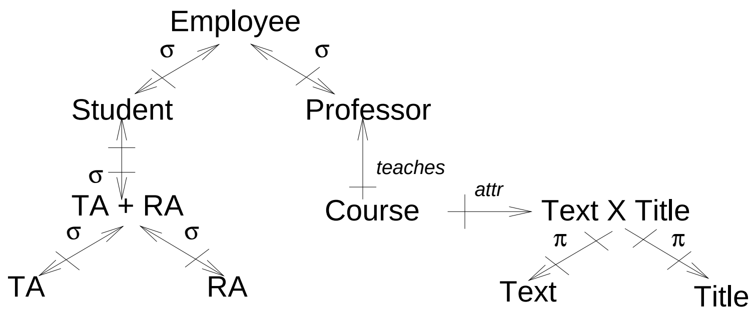


Figure 3: An example SIG schema.

3.3 Discussion of Model

The SIG model is inherently "data-centric" as opposed to "type-centric". By attaching a set of type definitions to an entity set, one can express constraints on the structure of individual entities. As noted above, however, our goal is to reason about constraints on *collections of entities* in an instance of the entity set, rather than about the internal structure of a single entity. Thus, instead of focusing on types and type level operations, we want to focus on instances and instance level operations.

To illustrate the difference between these approaches, consider a schema S1 containing sets of professors and sets of students (denoted by **Professors** and **Students**) and a second schema S2 containing the product of professors and students (denoted **Professors** × **Students**). The product constructor may be viewed as a type constructor or a set constructor. Under a "type-centric" view, the product constructor defines a new type; any professor–student pair is an element of this type, but the actual collection of pairs in a given instance can range from the empty set to the full cross-product of all professors and students.

Under our "data-centric" view, the product constructor operates on instances rather than types. Given instances of **Professors** and **Students**, an instance of **Professors** × **Students** is uniquely defined as the product of these sets. In the SIG model, cross product and union constructors are not viewed as defining new types (that is, new sets), but as combining existing sets. The motivation, as noted earlier, is to reason directly about the values in an (arbitrary) instance of the schema. For example, Schema S1 and Schema S2 are equivalent in the following sense: an instance of the first schema uniquely defines an instance of the second and vice versa. (This is clearly not true if × is viewed as a type constructor!) In Section 4, we formally define schema equivalence using as a basis the existence of such one-to-one correspondences between instances of schemas. Our model allows the expression of such equivalence preserving transformations as constructing the product or union of sets.

The SIG model also allows the definition of new sets that are only constrained to be subsets of existing sets, and so it can represent *complex types* built using union and cross-product (viewed as type constructors) [9]. While there is thus a connection between SIGs and complex types, there are important differences, as noted above.

4 Information Capacity

Formal notions of correctness for schema transformations are typically based on the preservation of the information content of schemas. We use a formalization in which the information content of a schema S is measured by the set of all valid instances of S, denoted $I(S)$. Such a formalization is appropriate for applications, like those described in Section 2, where one schema is used to store or access information in another. Two schemas can be compared based on information capacity. 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.[5]

Below we consider two formalizations of information capacity dominance and equivalence, namely absolute equivalence and internal equivalence. In our presentation, we analyze whether these definitions are appropriate for establishing results directly usable in the context of practical integration problems. For each notion, we present: (1) brief definitions; (2) existing results on testing for equivalence of various classes of schemas; (3) our results on testing for equivalence of SIGs; and (4) the potential use of this form of equivalence in practical systems.

4.1 Absolute Equivalence

Absolute equivalence gives a characterization of 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 (that is, injective) functions between the sets of instances of two schemas.

Definition 4.1 An *instance mapping* from schema $S1$ to $S2$ is any total function $f : I_Y(S1) \rightarrow I_Y(S2)$, where $Y \subseteq U$. •

Definition 4.2 An *information (capacity) preserving mapping* between the instances of $S1$ and $S2$ is a total, injective function $f : I_Y(S1) \rightarrow I_Y(S2)$. An *equivalence preserving mapping* between the instances of two schemas $S1$ and $S2$ is a bijection $f : I_Y(S1) \rightarrow I_Y(S2)$. •

Definition 4.3 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 $f : I_Y(S1) \rightarrow I_Y(S2)$. •

Work on characterizing absolute equivalence has focused on simple relational schemas and what has been termed "static schemas", that is types with no integrity constraints or dependencies. Characterizations of absolute dominance are known for relational schemas with only (primary) key dependencies [8]. For more complex static schemas, namely types formed by the recursive application of product, set or union constructors on infinite base types, absolute equivalence can be characterized by a set of natural restructuring operators [10]. The restructuring operators are used to define a normal form for these schemas such that two schemas are absolutely equivalent iff their normal forms are isomorphic. This result has been generalized to schemas that include finite as well as infinite base types [1]. Again, a (decidable) characterization of absolute equivalence for the extended set of schemas is given that is based on a set of restructuring operators. Characterization of absolute dominance for complex types are not known.

We now consider the problem of testing for absolute equivalence (or dominance) of SIGs. SIGs permit the representation of sets formed from nested product and union constructors, as well as simple constraints between these sets. The addition of these constraints makes testing for equivalence (and therefore dominance) of schemas undecidable. This result uses the fact that SIGs can express cardinality constraints on the relative size of nodes.

Lemma 4.1 Let $S = (G, \mathcal{A})$ be a SIG schema where $G = (N, E)$. Let $e : X - Y \in E$. Let $\Im$ be a valid instance of S. If $\mathcal{A}(e) \supseteq \{f, s\}$ then $|\Im[X]| \geq |\Im[Y]|$. •

Our undecidability result is a reduction from the problem of Diophantine equations. Let $\Theta(\vec{x})$ and $\Phi(\vec{x})$ be two polynomials with natural number coefficients over n variables (represented by $\vec{x}$). The equation $\Theta(\vec{x}) = \Phi(\vec{x})$ is referred to as a Diophantine equation and the problem of determining whether there exists a solution in the natural numbers is undecidable [6]. The next lemma states that Diophantine equations

[5]The form of relative information capacity we use in this work was first studied in [8] and [10]. Information capacity has also been applied to a number of translation and integration problems [1, 12, 15, 16, and others].

without constant terms may be "encoded" in annotated schema intension graphs. The details of the encoding and proof of the lemma are contained in Appendix A.

Lemma 4.2 Let $\Theta(\vec{x})$ and $\Phi(\vec{x})$ be two polynomials in n variables with no constant terms and with coefficients in $\mathcal{N}$, the natural numbers. Then, there exists a SIG schema S, called a *Diophantine encoding* of $\Theta(\vec{x}) = \Phi(\vec{x})$, containing nodes $X_1, X_2, ..., X_n$ such that the equation $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution $\vec{m} = (m_1, m_2, ..., m_n)$, $m_i \in \mathcal{N}$ iff there exists a valid instance $\Im$ for S where $|\Im[X_i]| = m_i$, $1 \leq i \leq n$. •

Theorem 4.3 Testing for absolute equivalence of SIGs is undecidable. •

Proof Let $\Theta(\vec{x})$ and $\Phi(\vec{x})$ be two polynomials (which may contain constant terms) and let w be a new variable. Let $S = (G, \mathcal{A})$ where $G = (N, E)$ be the Diophantine encoding of the equation $\Theta(\vec{x})w = \Phi(\vec{x})w$ and let the node W correspond to the variable w in the encoding.

Let $d : W - W$ be a new edge not in E. Let $G' = (N', E')$ where $N' = N$ and $E' = E \cup \{d\}$. Let $\mathcal{B}1$ be an annotation function on G' where $\mathcal{B}1(e) = \mathcal{A}(e)$ for all $e \in E$ and $\mathcal{B}1(d) = \emptyset$. Let $\mathcal{B}2$ be the annotation function on G' where $\mathcal{B}2(e) = \mathcal{A}(e)$ for all $e \in E$ and $\mathcal{B}2(d) = \{t\}$. Let $S1$ and $S2$ be the two SIG schemas $(G', \mathcal{B}1)$ and $(G', \mathcal{B}2)$ respectively.

By the definition of instances, every instance of $S2$ is an instance of $S1$, so for any $Y \subseteq U$, $I_Y(S2) \subseteq I_Y(S1)$.

It is easily verified that any valid instance $\Im$ of S can be extended to a valid instance of $\Im'$ of $S1$ by populating the edge d with any relation on the set $\Im[W]$ and that $\Im'$ is a valid instance of $S2$ iff $\Im'[d]$ is a total relation. Similarly, any valid instance of $\Im$ of $S1$ or $S2$ restricted to the nodes and edge of S is a valid instance of S.

We now prove that $S1 \not\sim_{abs} S2$ iff $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution. We use the fact that $S1 \sim_{abs} S2$ iff there exists some finite Z such that for all $Y \supseteq Z$ $|I_Y(S1)| = |I_Y(S2)|$.

($\Leftarrow$) Clearly, if $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution $\vec{m}$, then $\Theta(\vec{x})w = \Phi(\vec{x})w$ has solutions for which $w \neq 0$. By Lemma 4.2, there are therefore instances of S with $|\Im[W]| > 0$. Let $\Im$ be such an instance of S. Let $\Im'$ be an instance of $S1$ formed by extending $\Im$ with $\Im'[d] = \emptyset$. Hence, $\Im'$ is a valid instance of $S1$ but not a valid instance of $S2$ (since d is not total on $\Im[W]$). Suppose $S1 \sim_{abs} S2$, so for some finite Z, for all $Y \supseteq Z$, $|I_Y(S1)| = |I_Y(S2)|$. Let $Y = Sym(\Im') \cup Z$ so $\Im' \in I_Y(S1)$. Since $I_Y(S2) \subseteq I_Y(S1)$, the existence of $\Im'$ implies $|I_Y(S2)| < |I_Y(S1)|$. This contradicts the assumption that $S1 \sim_{abs} S2$. Hence, $S1 \not\sim_{abs} S2$.

($\Rightarrow$) Now suppose $S1 \not\sim_{abs} S2$. Then, for any finite Z, there exists some $Y \supseteq Z$ such that $|I_Y(S1)| \neq |I_Y(S2)|$. Since $I_Y(S2) \subseteq I_Y(S1)$, this implies there exists some $\Im \in I_Y(S1)$ where $\Im \notin I_Y(S2)$. The instance $\Im$ must populate d with a nontotal relation. If $|\Im[W]| = 0$ then there is no nontotal relation on $\Im[W]$. Therefore, $|\Im[W]| > 0$. The instance $\Im$ restricted to the nodes and edges of S is a valid instance of S and so by Lemma 4.2, $\Theta(\vec{x})w = \Phi(\vec{x})w$ has a solution for which $w \neq 0$. Therefore, $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution.

Since the problem of determining whether $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution is undecidable, testing for absolute equivalence of SIG schemas is also 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 [8]. This result, coupled with our undecidability result, show that absolute equivalence and dominance do not provide a sufficient foundation for analyzing practical integration problems.

4.2 Internal Equivalence

In an effort to overcome some of the limitations of absolute equivalence, various abstract properties have been proposed that restrict the class of allowable instance mappings [8]. Such restrictions include mappings that only reorganize and do not invent arbitrary values (termed *internal mappings*) and mappings that are queries in some query language. For internal equivalence and dominance, if two instances are associated via an instance mapping then they must contain (almost) the same set of symbols.

Definition 4.4 Let $Z \subseteq U$ be a finite set of data values. An information preserving mapping $f : I(S1) \rightarrow I(S2)$ is *Z-internal* if for all $\Im \in I(S1)$, $Sym(f(\Im)) \subseteq Sym(\Im) \cup Z$. The mapping f is said to be *internal* if it is Z-internal for $Z = \emptyset$. •

Definition 4.5 The schema $S2$ *internally dominates* $S1$, denoted $S1 \preceq_{int} S2$, if there exists a Z-internal information preserving mapping $f : I(S1) \rightarrow I(S2)$.[6] The schemas $S1$ and $S2$ are *internally equivalent*, denoted $S1 \sim_{int} S2$, if $S1 \preceq_{int} S2$ and $S2 \preceq_{int} S1$. •

Characterizations of internal dominance are known for relational schemas with only (primary) key dependencies [8]. For the classes of complex types considered in Section 4.1, internal equivalence is identical to absolute equivalence [1, 10]. Hence, the characterizations of absolute equivalence can be applied to determine if two schemas are internally equivalent. Decidable characterization of internal dominance for complex types are not known.

The use of internal mappings ensures that instances are only associated with instances containing the same symbols. However, this property is not sufficient to guarantee that internal mappings are well behaved. Schema instances with no intuitive relationship between them may still be associated under internal mappings [8]. Furthermore, internal equivalence suffers from the same problem as absolute equivalence in that testing for both internal equivalence and dominance of schemas is undecidable.[7]

Theorem 4.4 Testing for internal equivalence of SIGs is undecidable. •

Proof Consider $S1$ and $S2$ as defined in the proof of Theorem 4.3. Determining whether $S1 \sim_{abs} S2$ is undecidable. We now show that $S1 \sim_{abs} S2$ iff $S1 \sim_{int} S2$.

Suppose that $S1 \sim_{int} S2$. Since $S1 \preceq_{int} S2$, there exists some $\sigma : I(S1) \rightarrow I(S2)$ which is Z-internal for some Z. For each $Y \supseteq Z$, let σ_Y be the restriction of σ to $I_Y(S1)$. For each $\Im \in I_Y(S1)$, $\sigma(\Im) \in I_Y(S2)$, so $\sigma_Y : I_Y(S1) \rightarrow I_Y(S2)$. Since σ is a total injective function, so is each σ_Y. Hence, $|I_Y(S1)| \leq |I_Y(S2)|$ for each $Y \supseteq Z$. Since $S2 \preceq_{int} S1$, it is also that case that $|I_Y(S1)| \geq |I_Y(S2)|$ for each $Y \supseteq Z$. Hence, $S1 \sim_{abs} S2$. (In fact, this argument does not depend on the form of the schemas $S1$ and $S2$ so $S1 \sim_{int} S2 \Rightarrow S1 \sim_{abs} S2$ for arbitrary schemas [8].)

Conversely, suppose that $S1 \sim_{abs} S2$. By the definition of $S1$ and $S2$, every instance of $S2$ is an instance of $S1$. So, for any $Y \subseteq U$, $I_Y(S2) \subseteq I_Y(S1)$. Since $S1 \sim_{abs} S2$, there exists some finite Z such that $|I_Y(S1)| = |I_Y(S2)|$ for all $Y \supseteq Z$. Let $Y = U$, then $|I(S1)| = |I(S2)|$ and so $I(S1) = I(S2)$. Let $\sigma : I(S1) \rightarrow I(S2)$ be the identity relation. Then σ is a bijection, so in particular, σ and σ^{-1} are injective. Also, σ and σ^{-1} are Z-internal (for any Z). Hence, $S1 \sim_{int} S2$. So $S1 \sim_{abs} S2$ iff $S1 \sim_{int} S2$ and therefore testing for internal equivalence of SIG schemas is undecidable. □

5 Structural Transformations

Given our results that no decidable characterization of dominance or equivalence is possible using the given definitions of equivalence, the question remains as to how practitioners can develop rigorous methodologies. Our response is motivated by what practitioners currently do. Specifically, the more rigorous of the integration methodologies propose sets of equivalence preserving transformations that may be used in translating schemas between or within data models [7, 11, 12, 16, and others]. These transformations have proven to be successful in automating the translation of both schemas and instances. If it is not possible to test for dominance of two schemas through an arbitrary abstract instance mapping, perhaps it is possible to test for dominance through an instance mapping created by some predefined set of transformations.

In this section, we take this approach by considering several schema transformations for SIGs. In understanding the transformations presented, it is important to note that unlike transformations presented elsewhere, we are interested in transformations that preserve equivalence or transformations that augment

[6]This definition of internal dominance corresponds to *internal embeddability.* However, the proof that these two notions are equivalent for relational schemas can be extended to SIGs [8].

[7]Our result shows that even testing for internal dominance through an internal mapping (where $Z = \emptyset$) is undecidable. This is actually a stronger condition than general internal dominance.

information capacity (that is, transformations that only preserve dominance) [1, 10, and others].[8] We define measures of dominance that are *complete* with respect to these transformations and form sufficient conditions for internal dominance. The importance of these characterizations comes from the fact that each leads to a procedure for testing if one schema can be produced from another through any sequence of these transformations. This procedure can also generate an instance mapping, which is crucial for a schema transformation to be useful in a practical environment. We conclude this section with a discussion of the complexity of algorithms for deciding dominance.

In general, a transformation T defines a function on sets of schemas $T : \mathcal{S}_1 \rightarrow \mathcal{S}_2$. The transformations we consider are defined on the class of all SIG schemas. For schemas $S1$ and $S2$, $T(S1) = S2$ is denoted by $S1 \xrightarrow{T} S2$. An arbitrary (possibly empty) sequence of transformations $\xrightarrow{T}$ is denoted $\xrightarrow{T}^{*}$. If X is a set of specific transformations then $\vec{X}$ denotes a sequence containing all the transformations in X.[9]

In any natural characterization of equivalence, it must be the case that isomorphic schemas are equivalent. We therefore begin by considering isomorphism as the basis for determining equivalence of schemas. SIG isomorphism is a special case of graph isomorphism constrained to preserve the types of nodes and all constraints placed on edges (recall that SIG constraints include annotations as well as projection and selection constraints).[10]

5.1 Annotation Transformations

An annotation transformation (or α-transformation) allows the removal of annotations or projection and selection constraints from an edge of a schema. An example is shown in Figure 4.

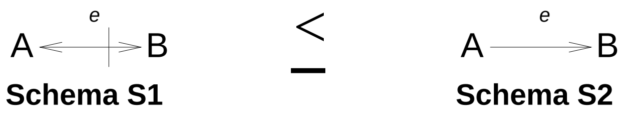


Figure 4: An α-transformation.

Definition 5.1 Let $S1 = (G1, \mathcal{A}1)$ and $S2 = (G2, \mathcal{A}2)$ be two SIG schemas. Let $G1 = G2$ and $\mathcal{A}1 = \mathcal{A}2$ except $\mathcal{A}1(e) \supseteq \mathcal{A}2(e)$ for some edge e and if e is a selection (projection) edge in $G2$ then it is a selection (projection) in $G1$. Then, $S1 \xrightarrow{\alpha_e} S2$ and α_e is called an *annotation transformation (α-transformation).*[11] •

If $S1 \xrightarrow{\alpha} S2$ then every instance of $S1$ is also an instance of $S2$. Hence, the identity function on $I(S1)$ is an information preserving mapping from $S1$ to $S2$ and we have the following immediate result.

Theorem 5.1 If $S1 \xrightarrow{\alpha} S2$ then $S1 \preceq_{int} S2$. •

5.2 Composition Transformations

Consider again the relational example presented in Section 2. There we had a unique **Grant** associated with every **Project** and wanted to determine if it was possible to move the **Grant** attribute to the **Workstation** table without losing information capacity. We argued informally that if every **Workstation** determined a unique **Project** this would indeed be the case. Figure 5 depicts such a scenario in SIG form. If we have a functional edge (or path) from **Workstation** to **Project**, then we can move the **Grant** attribute across the path. The edge g in Schema $S2$ from **Workstation** to **Grant**, can be populated with instances of the path $e \circ p$ from Schema $S1$.

[8]Work on hierarchical data structures includes some consideration of transformations that increase information capacity, but only characterizes equivalence (and not dominance), under these transformations [1].

[9]We use this notation when the transformations in X commute so that there is a unique schema $S2$ such that $S1 \xrightarrow{\vec{X}} S2$.

[10]SIG isomorphism is defined precisely in Appendix B.

[11]When the edge e is understood from context (or not relevant to the discussion) it may be omitted and the α-transformation denoted α.

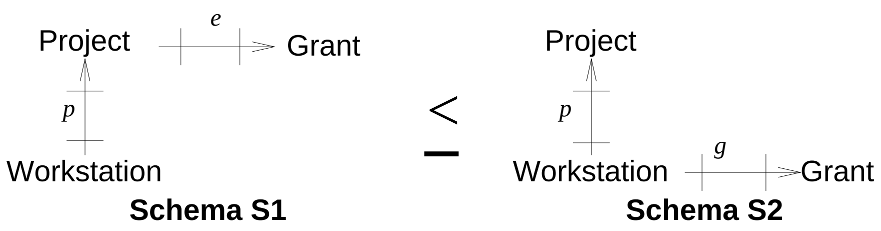


Figure 5: Moving an attribute.

5.2.1 Definitions

Using the intuition provided by this example, we consider the general case of how an edge of a schema may be "encoded" by an edge between different nodes. We want to ensure that any such transformation induces information preserving instance mappings between edges. Consider the schemas of Figure 6. Intuitively, instances of Schema $S1$ can be mapped to instances of Schema $S2$ by populating the edge g with the result of composing p with e. For this to be an information preserving transformation, however, it must be the case that every instance of the edge e determines a unique instance of g so that an instance of e is recoverable from its image under the transformation. Lemma 5.2 defines precisely when this is the case.

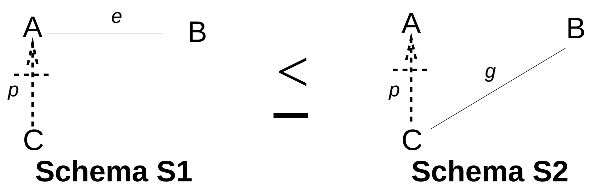


Figure 6: A $\circ$-transformation.

Lemma 5.2 Let $\Im[A]$, $\Im[B]$, and $\Im[C]$ be finite sets. Let $\Im[p] : \Im[C] - \Im[A]$ be a relation.

1. If $\Im[p]$ is a surjective function, then for all relations $\Im[e] : \Im[A] - \Im[B]$, there exists a unique relation $\Im[g] : \Im[C] - \Im[B]$ such that $\Im[g] = \Im[e] \circ \Im[p]$.

2. If $|\Im[B]| > 1$, then for all $\Im[e] : \Im[A] - \Im[B]$, there exists a unique relation $\Im[g] : \Im[C] - \Im[B]$ such that $\Im[g] = \Im[e] \circ \Im[p]$, only if $\Im[p]$ is a surjective function. •

Proof The proof uses simple algebraic reasoning. □

Using this result, we can encode an instance of an edge $e : A - B$ in an edge $g : C - B$ that shares a common end node, so long as there is a surjective functional path from C to A. Furthermore, Lemma 5.2 indicates that this encoding (using the composition) is only possible if the path is a surjective function or some set of constraints serve to restrict the set of all valid instances of B to sets of size 1 or 0. Since determining this latter property is undecidable [14], we restrict our transformations to use only the first property.

We now examine if it is possible to encode the edge e in an edge $g : C - D$ with two different end points. In Figure 7, we show that Lemma 5.2 can be applied twice to encode $e : A - B$ in the edge $g : C - D$. This motivates the following definition of composition transformations.

Definition 5.2 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 $S1 \xrightarrow{\circ^e_g} S2$ is called a *simple composition transformation* (a *simple $\circ$-transformation*).[12] •

[12]Again, the edges e or g may be omitted and the $\circ$-transformation may be denoted $\circ$ ($\circ^e$ or $\circ_g$).

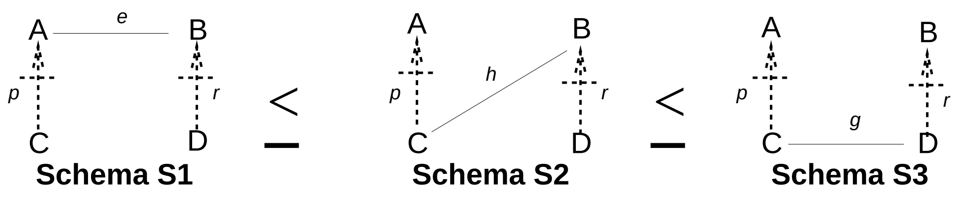


Figure 7: Two $\circ$-transformations.

Theorem 5.3 Let $\circ^e_g$ be a simple $\circ$-transformation that uses the surjective functional paths p and r. If $S1 \xrightarrow{\circ^e_g} S2$ then $S1 \preceq_{int} S2$. If $\mathcal{A}1(p) = \mathcal{A}1(r) = \{f, i, s, t\}$ and the constraints of g are equal to the constraints of e then $S1 \sim_{int} S2$. •

Proof We use an instance mapping $f : I(S1) \rightarrow I(S2)$ that is the identity everywhere except $f(\Im)[g] = \Im[r^\circ] \circ \Im[e] \circ \Im[p]$. The binary relation $f(\Im)[g]$ satisfies all constraints satisfied by each of e, p and r°, which by Definition 5.2 are the same as the constraints on g. Hence, $f(\Im)$ defines a valid instance of $S2$. Also, f is clearly a total function and is internal.

By Lemma 5.2, given an instance of p, for any instance of e there is a unique instance of $e \circ p$. Applying the lemma a second time, for any instance of $e \circ p$ and r°, there is a unique instance of $r^\circ \circ e \circ p$. Hence, the instance mapping f is injective. Therefore, f is information preserving and $S1 \preceq_{int} S2$.

Assume that p and r are bijective paths and the constraints of g are equal to the constraints of e. Then, for any instance of g, $r \circ g \circ p^\circ$ is a valid instance of e so the mapping f is surjective. Hence, the mapping f is bijective and therefore equivalence preserving and so $S1 \sim_{int} S2$. □

While we would like to say that a $\circ$-transformation is equivalence preserving *iff* $\mathcal{A}1(p) = \mathcal{A}1(r) = \{f, i, s, t\}$, this may not be true. The problem of annotation implication is undecidable for SIGs [13]. Hence, it may be the case that $\mathcal{A}1(p) \subset \{f, i, s, t\}$ and yet the only valid instances of p are bijections. In this case, the $\circ$-transformation could still be equivalence preserving.

We now consider sequences of simple $\circ$-transformations. The application of a $\circ$-transformation cannot create new surjective functional paths. More precisely, if $S1 \xrightarrow{\circ}{}^{*} S2$ and there is a surjective functional path $p : A{-}B$ in $S2$, then there is some surjective functional path $p' : A{-}B$ in $S1$. Clearly, a $\circ$-transformation may destroy surjective functional paths. Hence, there may be sets of transformations that cannot be serialized. For example, there may be two transformations, each of which can be applied to a given schema. Once either transformation is applied though, the second transformation cannot be applied to the transformed schema because one or both of the surjective functional paths required by the transformation has been removed. However, by applying both transformations in "parallel", we can still construct a meaningful information preserving instance mapping. We therefore generalize the definition of $\circ$-transformations to allow simple $\circ$-transformations to be applied in parallel. This definition permits a broader class of transformations.

Definition 5.3 A $\circ$-*transformation* is a set of one or more simple $\circ$-transformations. A $\circ$-transformation is denoted $S1 \xrightarrow{\{\circ^{e1}_{g1}, \ldots \circ^{en}_{gn}\}} S2$ where the $\circ^{ei}_{gi}$ are simple $\circ$-transformations and all g_i are distinct. For the case $n = 1$, the braces may be omitted, $S1 \xrightarrow{\circ^e_g} S2$. •

Theorem 5.4 Let $S1 \xrightarrow{\{\circ^{e1}_{g1}, \ldots \circ^{en}_{gn}\}} S2$. Then, $S1 \preceq_{int} S2$. If each component simple $\circ$-transformation is equivalence preserving and all e_i are distinct then $S1 \sim_{int} S2$. •

Proof Let $O = \{\circ^{e1}_{g1}, \ldots \circ^{en}_{gn}\}$. Again, we use an instance mapping $f : I(S1) \rightarrow I(S2)$ that is the identity everywhere except for each $\circ^{ei}_{gi} \in O$, $f(\Im)[g_i] = \Im[r^\circ_i] \circ \Im[e_i] \circ \Im[p_i]$. For each i, the binary relation $f(\Im)[g_i]$ satisfies all constraints satisfied by each of e_i, p_i and r°_i, which by Definition 5.2 are the same as the annotations on g_i. Hence, $f(\Im)$ defines a valid instance of $S2$. Also, f is a total function. By Lemma 5.2, for any instance of e_i, p_i and r°_i there is a unique instance of $r^\circ_i \circ e_i \circ p_i$. Hence, the instance mapping f is injective. Therefore, f is information preserving. The mapping f is also internal so $S1 \preceq_{int} S2$.

Assume that all p_i and r_i are bijective paths. Then, for any instance of g_i, $r_i \circ g_i \circ p_i^\circ$ is a valid instance of e_i. Since each e_i is distinct, every edge of $S1$ corresponds to exactly one edge of $S2$ so f is surjective. Hence, the mapping f is bijective and therefore equivalence preserving. So, $S1 \sim_{int} S2$. □

By permitting the application of simple $\circ$-transformations in parallel, we also allow a single edge e of $S1$ to be mapped to multiple edges of $S2$. This permits the transformation of Schema $S1$ of Figure 8 into Schema $S2$. The edge *work-addr* is mapped both to itself and to the edge *home-addr*. Intuitively, instances of $S1$ are a subset of the instances of $S2$ where *home-addr* is populated with the work address of each employee and people who are not employees have no home address. Clearly, $S1$ and $S2$ are not equivalent under this mapping since there are instances of $S2$ that populate *work-addr* and *home-addr* with different binary relations (and that assign unemployed people home addresses).

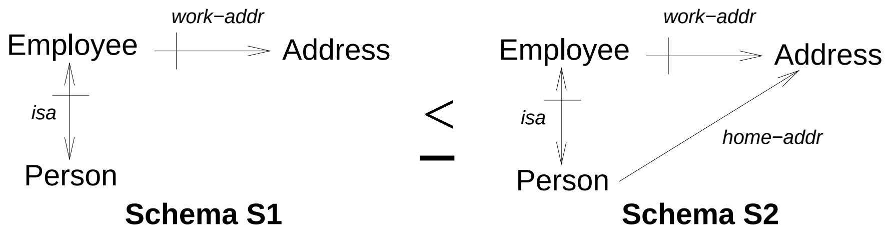


Figure 8: A $\circ$-transformation mapping *work-addr* to *work-addr* and *home-addr*.

The composition of the information preserving mappings created by $\circ$-transformations and α-transformations is also information preserving. The notation $\xrightarrow{\alpha\circ}{}^*$ indicates an arbitrary sequence of α-transformations or $\circ$-transformations.

Corollary 5.5 If $S1 \xrightarrow{\alpha\circ}{}^* S2$ then $S1 \preceq_{int} S2$. •

Several researchers have made use of information or equivalence preserving transformations that move or copy attributes between entities or classes within other data models [7, 17]. These transformations are special cases of $\circ$-transformations. However, no complete characterization of dominance under these transformations has been given. Hence, it was not previously possible to determine if an arbitrary schema could be transformed into another via these transformations.

5.2.2 Characterization of Dominance

We now define a notion of dominance for SIGs that is complete with respect to both α-transformations and $\circ$-transformations. Specifically, we define $\alpha\circ$-dominance ($\preceq_{\alpha\circ}$) such that $S1 \xrightarrow{\alpha\circ}{}^* S2$ iff $S1 \preceq_{\alpha\circ} S2'$ for some $S2'$ that is isomorphic to $S2$. The definition uses node and edge maps (defined in Appendix B), which are binary relations defined to preserve node construction and edge inversion, respectively.

Definition 5.4 Let $S1$ and $S2$ be two SIG schemas. Then, $S2$ $\alpha\circ$-*dominates* $S1$, denoted $S1 \preceq_{\alpha\circ} S2$, if there exist a bijective node map $\psi : N1 - N2$ and a total, surjective, injective edge map $\theta : E1 - E2$ satisfying the following: [13]

- 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 $r : D - B$ (not containing e) in $S1$ and
- the constraints on g' in $S2$ are a subset of the constraints on the path $r^\circ \circ e \circ p$ in $S1$.

Definition 5.5 Let $S1$ and $S2$ be two SIG schemas. Then, $S1$ is $\alpha\circ$-*equivalent* to $S2$, denoted $S1 \sim_{\alpha\circ} S2$, if $S1 \preceq_{\alpha\circ} S2$ and $S2 \preceq_{\alpha\circ} S1$. •

[13]To add clarity to the proofs, when the schemas $S1$ and $S2$ are arbitrary SIGs, we use the convention that edges and nodes in $S2$ are denoted by primed symbols (for example, e' and A') while edges and nodes in $S1$ are denoted by nonprimed symbols. When $S2$ is created by a series of transformations from $S1$, we may also find it convenient to denote nodes and edges in $S2$ that are not also in $S1$ by primed symbols.

Before proving that $\alpha\circ$-dominance completely characterizes schemas that can be obtained through a sequences of α-transformations and $\circ$-transformations, we state two lemmas that will be used in the proof. The proofs of these lemmas are in Appendix C. The first lemma states that if a sequence of $\circ$-transformations creates an edge then there exists a single $\circ$-transformation on the original schema that creates an edge between the same nodes with the same constraints. The second lemma states that any α-transformation can be "pushed to the right" past $\circ$-transformations.

Lemma 5.6 Let $S1 \xrightarrow{\circ}{}^{*} S2$ where $S2$ contains the edge $g' : C - D$. Then, there exist an edge $e : A - B$ and surjective functional paths $p : C - A$, $r : D - B$ (not containing e) in $S1$ such that the constraints on g' in $S2$ are exactly the constraints on the path $r^\circ \circ e \circ p$ in $S1$. •

Lemma 5.7 If $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ then there exists an $S1'$ such that $S1 \xrightarrow{\circ}{}^{*} S1' \xrightarrow{\alpha}{}^{*} S2$. •

Theorem 5.8 Let $S1$ and $S2$ be two SIG schemas. Then, $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ iff $S1 \preceq_{\alpha\circ} S2'$, where $S2' \cong S2$. •

Proof ($\Leftarrow$) We first show that if $S1 \preceq_{\alpha\circ} S2'$ then $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ for some $S2 \cong S2'$.

Let ψ and θ be specific node and edge maps satisfying the definition of $\alpha\circ$-dominance. Let $g' : \psi(C) - \psi(D)$ be an edge of $S2'$ and let $e : A - B$ be the edge of $S1$ such that $g' \in \theta(e)$ (e exists and is unique since θ is surjective and injective). We define $\circ_g^e$ to be the transformation carrying e to $g : C - D$. By the definition of $\preceq_{\alpha\circ}$, there must be surjective functional paths $p : C - A$ and $r : D - B$ in $S1$ and so $\circ_g^e$ is a valid transformation that creates an edge g with exactly the constraints on the path $r^\circ \circ e \circ p$. We also define $\alpha_{e,g}$ to be the α-transformation that removes from g all annotations on the path $r^\circ \circ e \circ p$ that are not on the edge g'. By the definition of $\alpha\circ$-dominance, the constraints on g' are a subset of the constraints on this path so this is a valid α-transformation.

Let $\circ_E$ be the $\circ$-transformation containing all $\circ$-transformations created in this manner, $\circ_E = \{\circ_g^e \mid e \in E1$ and $g' \in \theta(e)\}$ and let A be the set of all α-transformations created. Let $S2$ be a SIG schema that results from the application of $\circ_E$ and A to $S1$, $S1 \xrightarrow{\circ_E} T \xrightarrow{\vec{A}} S2$. Let $\jmath : E2 - E2'$ be an edge map such that for all $g \in E2$, $\jmath(g) = g' \in E2'$. Since θ is total, every edge g of $S2$ is created by some (possibly trivial) $\circ$-transformation $\circ_g^e$ and so $\jmath$ is a bijective edge map. The nodes of $S2$ are the same as the nodes of $S1$ so ψ is a bijective node map on the nodes of $S2$ and $S2'$. The schema $S2$ is isomorphic to $S2'$ via the edge bijection $\jmath$ and node bijection ψ.

($\Rightarrow$) Next, we show that if $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ then $S1 \preceq_{\alpha\circ} S2$ by constructing maps ψ and θ.

The α-transformations and $\circ$-transformations do not create or delete nodes so $N1 = N2$. Let ψ be the identity on $N1$.

Since $S1 \xrightarrow{\alpha\circ}{}^{*} S2$, by Lemma 5.7, there exists a sequence of transformations such that $S1 \xrightarrow{\circ}{}^{*} S1' \xrightarrow{\alpha}{}^{*} S2$. Since α-transformations do not remove or add edges, $E1' = E2$. Let $E = E2 - E1$ be the edges of $S2$ created by the $\circ$-transformations. For each $g \in E$, by Lemma 5.6, there exists an edge $e : A - B$ and surjective functional paths $p : C - A$, $r : D - B$ in $S1$. Since the α-transformations only remove constraints, it must be the case that the constraints on g in $S2$ are a subset of the constraints on the path $r^\circ \circ e \circ p$ in $S1$. We therefore let $g \in \theta(e)$ (and $g^\circ \in \theta(e^\circ)$). All other edges $e \in E1$ are in $E2$ so let $e \in \theta(e)$. For these edges, the paths p and r are the trivial paths and clearly, the constraints on e in $S2$ are a subset of the constraints on e in $S1$. The map θ is defined on all edges of $E1$ and is therefore total. Since every edge of $S2$ must exist in $S1$ or be created by some $\circ$-transformation, θ is also surjective. Additionally, by its definition θ is an injective edge map. Hence, θ satisfies the requirements of Definition 5.4 and $S1 \preceq_{\alpha\circ} S2$ via θ and the identity node map.

The composition of any SIG isomorphism (that is, the node and edge maps of an isomorphism) with the maps ψ and θ also satisfies Definition 5.4 and so if $S2 \cong S2'$ then $S1 \preceq_{\alpha\circ} S2'$, as required. □

By Theorem 5.8 and Corollary 5.5, we obtain the following result.

Corollary 5.9 If $S1 \preceq_{\alpha\circ} S2$ then $S1 \preceq_{int} S2$. If $S1 \sim_{\alpha\circ} S2$ then $S1 \sim_{int} S2$. •

5.2.3 Testing for Dominance

The importance of Definition 5.4 is that it leads immediately to an algorithm for determining when two arbitrary SIG schemas are in an $\alpha\circ$-dominance relation. By Theorem 5.8, $\alpha\circ$-dominance holds precisely when a schema may be obtained from another through some sequence of α-transformations and $\circ$-transformations. Hence, it is not necessary to consider all possible sequences of α-transformations and $\circ$-transformations. The monotonic nature of α-transformations ensures that there are only a finite number of possible sequences of (nontrivial) α-transformations that apply to a given schema. However, the definition of $\circ$-transformations permits an infinite number of sequences. We are only able to determine whether $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ because the decidable characterization of Definition 5.4 is complete with respect to α-transformations and $\circ$-transformations. Furthermore, we have both a sufficient condition for internal dominance and an algorithm testing this condition.

The proof of Theorem 5.8 is constructive. That is, if $S1 \preceq_{\alpha\circ} S2$ we can produce both a sequence of α-transformations and $\circ$-transformations such that $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ and the instance mappings induced by these transformations.

Corollary 5.10 If $S1 \preceq_{\alpha\circ} S2$ then we can construct a sequence of α-transformations and $\circ$-transformations such that $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ via this sequence and an information preserving mapping $f : I(S1) \rightarrow I(S2)$. •

As demonstrated by the examples of Section 2, being able to generate the instance mappings between schemas as in the above corollary is important. Furthermore, the actual transformation sequence can be used to explain to a designer the reason two schemas are equivalent or in a dominance relation. We address the complexity of the construction in Section 5.5.

5.3 Selection Transformations

The introduction of $\circ$-transformations allowed us to consider mappings of edges to multiple edges in a transformed schema. However, $\alpha\circ$-dominance still requires a bijection on nodes. We now consider transformations that permit mappings of nodes to multiple nodes.

5.3.1 Definitions

Selection transformations (ς-transformations) are transformations of the form shown in Figure 9. They permit the creation of new nodes and edges.

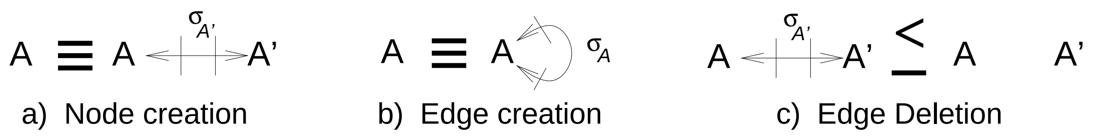


Figure 9: Selection transformations.

A *node creation ς-transformation* creates a new node that is isomorphic to an existing node (Figure 9a). A bijective selection edge between the two nodes enforces the constraint that the nodes be assigned identical sets in any valid instance.

Definition 5.6 Let A be a node of schema $S1$. Let A' be a new node not in $S1$ and $\sigma_{A'} : A \leftrightarrow A'$ a new bijective selection edge. Let $S2$ be $S1$ with the addition of the node A' and the edge $\sigma_{A'}$. 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.* •

Note that we only permit the removal of a node A' (and the incident bijective selection edge) when the selection edge is the only edge incident to A'. However, any edges incident to A' can be moved (using $\circ$-transformations) across the selection edge to the node A before removing A'.

In addition to transformations that add and remove nodes, we consider transformations that add and remove edges. An *edge creation ς-transformation* creates a new edge (Figure 9b). To preserve information capacity, the new edge is a bijective selection edge on a node. Note that $\circ$-transformations and α-transformations can subsequently be used to move the edge or remove the constraints on the new edge. An *edge deletion ς-transformation* removes an edge (Figure 9c). 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. If a bijective selection edge $\sigma_A : A \leftarrow\!\!+\!\!+\!\!\rightarrow A'$ is removed, information capacity may be augmented since the constraint that A and A' be assigned the same set in any valid instance may have been removed. However, if $A = A'$ then information capacity is preserved.

Definition 5.7 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 5.11 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, ς is an edge deletion ς-transformation that removes an edge between distinct nodes and $S1 \preceq_{int} S2$. •

Proof Let $S1 \xrightarrow{\varsigma_A} S2$ be a node creation ς-transformation and $S2 \xrightarrow{\varsigma_A} S1$ be a node deletion ς-transformation. Let $f : I(S1) \rightarrow I(S2)$ be an instance mapping that is the identity on all nodes and edges common to $S1$ and $S2$ and let $f(\Im)[A'] = \Im[A]$ and $f(\Im)[\sigma_{A'}] = 1_A$. The mapping f creates a valid instance of $S2$ and is bijective and internal. Hence, $S1 \sim_{int} S2$.

Let $S1 \xrightarrow{\varsigma_{\sigma_A}} S2$ be an edge creation ς-transformation. Let $f : I(S1) \rightarrow I(S2)$ be an instance mapping that is the identity on all nodes and edges common to $S1$ and $S2$ and let $f(\Im)[\sigma_A] = 1_A$. The mapping f creates a valid instance of $S2$ and is bijective and internal. Hence, $S1 \sim_{int} S2$.

Let $S1 \xrightarrow{\varsigma_{\sigma_A}} S2$ be an edge deletion ς-transformation where $\sigma_A : A - A'$. Let $f : I(S1) \rightarrow I(S2)$ be an instance mapping that is the identity on nodes and edges common to $S1$ and $S2$. The mapping f creates a valid instance of $S2$ and is a total, injective, function and internal. Hence, $S1 \preceq_{int} S2$. If $A = A'$ then the mapping f is surjective and so $S1 \sim_{int} S2$. (If $A \neq A'$, the mapping f is not necessarily surjective since there may be valid instances of $S2$ in which A and A' are not populated with the same set.) □

Corollary 5.12 If $S1 \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ then $S1 \preceq_{int} S2$. •

Selection transformations are of limited use in isolation. However, when combined with $\circ$-transformations and α-transformations, they permit complex additions and modifications to be made to a schema. An example is given in Section 5.4.

5.3.2 Characterization of Dominance

We now develop a characterization of dominance that is complete with respect to all transformations considered, α-transformations, $\circ$-transformations, and ς-transformations.

In the definition of $\alpha\circ\varsigma$-dominance, there are three main changes from the definition of $\alpha\circ$-dominance. First, the node map is no longer required to be a function. For a node $A \in N1$, if $A', B' \in \psi(A)$ then B' is the image of a node created by a node ς-transformation from A. Second, the node and edge maps are no longer required to be total. If a map is not defined on a node or edge of $S1$, then (intuitively) this node or edge has been removed by an ς-transformation. Finally, an edge of $E2$ may map via θ^{-1} to an edge of $E1$ (as before) or to a node of $N1$. This latter case indicates that in the sequence of transformations creating $S2$ from $S1$, this edge of $E2$ is created by a ς-transformation. In an instance mapping between $S1$ and $S2$, such an edge will be populated with the identity relation.

Definition 5.8 Let $S1$ and $S2$ be two SIG schemas. Then, $S2$ $\alpha\circ\varsigma$-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 $r : D - B$ (not containing e) in $S1$ and the constraints on g' in $S2$ are a subset of the constraints on the path $r^{\circ} \circ e \circ p$ in $S1$. •

Definition 5.9 Let $S1$ and $S2$ be two SIG schemas. Then, $S1$ is $\alpha\circ\varsigma$-*equivalent* to $S2$, denoted $S1 \sim_{\alpha\circ\varsigma} S2$, if $S1 \preceq_{\alpha\circ\varsigma} S2$ and $S2 \preceq_{\alpha\circ\varsigma} S1$. •

We now state the main result of this section that $\alpha\circ\varsigma$-dominance is complete with respect to all transformations considered, α-transformations, $\circ$-transformations, and ς-transformations. The proof is rather involved and is included in Appendix D along with a number of lemmas used in the proof.

Theorem 5.13 Let $S1$ and $S2$ be two SIG schemas. Then, $S1 \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$ iff $S1 \preceq_{\alpha\circ\varsigma} S2'$, where $S2' \cong S2$. •

By Theorem 5.13 and Corollary 5.12, we obtain the following result.

Corollary 5.14 If $S1 \preceq_{\alpha\circ\varsigma} S2$ then $S1 \preceq_{int} S2$. If $S1 \sim_{\alpha\circ\varsigma} S2$ then $S1 \sim_{int} S2$. •

5.3.3 Testing for Dominance

Definition 5.8 essentially gives an algorithm to test if two SIG schemas are in an $\alpha\circ\varsigma$-dominance relation, which by Theorem 5.13, also determines if a schema may be obtained from another through any sequence of transformations. As was the case for $\alpha\circ$-dominance, if $S1 \preceq_{\alpha\circ\varsigma} S2$ we can produce both a sequence of transformations such that $S1 \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$ and the instance mappings induced by these transformations.

Corollary 5.15 If $S1 \preceq_{\alpha\circ\varsigma} S2$ then we can construct a sequence of α-transformations, $\circ$-transformations and ς-transformations such that $S1 \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$ via this sequence and an information preserving mapping $f : I(S1) \rightarrow I(S2)$. •

5.4 An Example

The example of Figure 10 illustrates the generality of the transformations we have considered. A designer wishes to determine if an arbitrary instance of $S1$ can be transformed into an instance of $S2$ without losing any information. On the surface, the schemas appear to be quite dissimilar and no immediate instance level mapping is evident.

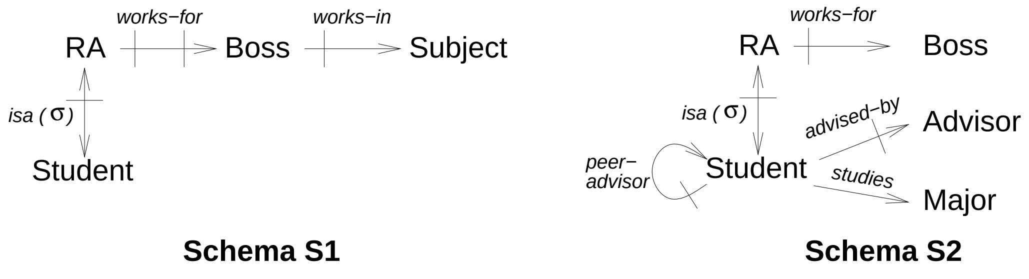
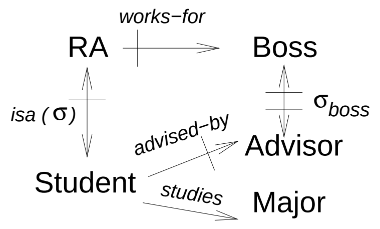


Figure 10: Does Schema $S2$ dominate Schema $S1$?

We can use our characterization of $\alpha\circ\varsigma$-dominance to determine that $S1 \preceq_{int} S2$ and to produce the instance level mapping. Suppose we are given the typing information that nodes with like names have the same type and further that $\tau(Subject) = \tau(Major)$ and $\tau(Boss) = \tau(Advisor)$. We can produce the following mappings ψ and θ satisfying Definition 5.8.

$$\begin{array}{ll} \psi(RA) = \{RA\} & \psi(Student) = \{Student\} \\ \psi(Boss) = \{Boss, Advisor\} & \psi(Subject) = \{Major\} \\ \theta(isa) = \{isa\} & \theta(worksfor) = \{worksfor, advisedby\} \\ \theta(worksin) = \{studies\} & \theta(Student) = \{peeradvisor\} \end{array}$$

These mappings correspond to the following sequence of transformations. A node creation ς-transformation, $\varsigma_{Advisor}$, creates the node *Advisor* and a bijective selection edge σ_{Boss} between *Boss* and *Advisor*. The edge *worksfor* is copied to *worksfor* and to *advisedby* (across the paths $p = isa$ and $r = \sigma_{Boss}$). The edge *worksin* is moved to the edge *studies* across the path $p = worksfor \circ isa$. The result of these transformations is the intermediate schema depicted in Figure 11. Next an edge creation ς-transformation, $\varsigma_{Student}$, creates the bijective selection edge *peeradvisor* on *Student*. An α-transformation, $\alpha_{peeradvisor}$, removes the selection, surjectivity and injectivity constraints on the edge. Another α-transformation, $\alpha_{worksfor}$, removes the surjectivity constraints on the *worksfor* edge. Finally, an edge deletion ς-transformation, $\varsigma_{\sigma_{Boss}}$, removes the edge σ_{Boss}. The result of these transformations is the schema $S2$.



Intermediate Schema T

Figure 11: Intermediate schema $S1 \overset{\alpha\circ\varsigma^*}{\longrightarrow} T \overset{\alpha\circ\varsigma^*}{\longrightarrow} S2$.

To automatically translate queries on $S2$ into queries on $S1$, the instance mapping $f : I(S1) \rightarrow I(S2)$ may be produced from the maps ψ and θ. A portion of f is show below.[14]

$$\begin{array}{ll} f(\Im)[Advisor] = \Im[Boss] & f(\Im)[Boss] = \Im[Boss] \\ f(\Im)[studies] = \Im[worksin] \circ \Im[workfor] \circ \Im[isa] & f(\Im)[peeradvisor] = 1_{\Im[Student]} \end{array}$$

There may be more than one possible instance mapping (produced by different sequences of transformations). Our algorithms can automatically produce all valid instance mappings. However, the algorithms, necessarily, perform only syntactic analysis of the schemas. They may take advantage of semantic information (such as node types) provided by the designer but they can never fully automate a designer's semantic understanding of a schema.

In the example presented, the *peeradvisor* edge of Schema $S2$ is populated with the identity relation on *Student*. Such a mapping is valid but may not make semantic sense. Can a student advise herself or himself? If typing information on edges is provided, such mappings may be avoided. For this example, a designer may declare the type of *worksfor* to be the same as *advisedby* but the type of *peeradvisor* different from $1_{Student}$. In general, it may not be possible to specify a priori such typing information. Furthermore, not all semantic information may be conveyed by types on edges or nodes.

What a tool provides is an automatic way of testing for information capacity preservation. It relieves a designer of the burden of having to verify that for each instance of a schema, a unique instance of another may be created without loss of information. It cannot, however, fully understand the semantics of arbitrary schemas. Rather than trying to automate semantic understanding of schemas, our results permit the development of tools that identify when instances may be rearranged in a certain fashion. The final decision on whether the rearrangement makes semantic sense must always come from a designer.

[14]The function $1_{\Im[Student]}$ is the identity relation on the set $\Im[Student]$.

5.5 Complexity of Dominance Algorithms

If every node of both schemas has the same type and every edge has the same set of constraints then T-dominance[15] holds precisely when there exists an isomorphism on the underlying graphs of the schemas. There are no known algorithms for graph isomorphism in polynomial time and it is an open problem whether this problem is **NP-complete**. Thus, the best (worst case) bound for T-dominance we have is **NP** (nondeterministic polynomial time). This result uses a simple modification to graph isomorphism.

However, such worst case behavior is of theoretical rather than practical importance. The schemas that arise in practice do not exhibit this worst case behavior. In particular, they are formed from nodes of different abstract types with edges having diverse combinations of constraints. Both of these facts can be used to reduce the number of possible mappings between schemas that are considered by an algorithm. Additionally, information about the structure of common schemas can be used to produce algorithms that are efficient on such schemas. Below, we briefly examine the process of generating both the node maps and edge maps required by the definitions of T-dominance. While space prohibits the presentation of a full algorithm, we mention some key points that enable the development of efficient algorithms.

5.5.1 Node Maps

There are typically only a few nodes of the same type in a schema and these nodes are often related in a tree structure (through an inheritance hierarchy represented in SIGs by selection edges). Even in the absence of an inheritance hierarchy the annotations on edges permit us to define a partial order on the nodes of a graph.

Definition 5.10 Let S be a SIG schema. For nodes $A, B \in N$, A is *node dominated by* B, denoted $A \ll B$, if there exists a surjective functional path from B to A. •

The relation $\ll$ defines a partial order on equivalence classes of nodes.[16] Furthermore, in any valid instance of S, the node B must be assigned at least as many elements as A. We can use this fact to prove the following theorem about any node map ψ used to show that $S1 \preceq_T S2$.

Theorem 5.16 If $A' \ll B'$ in $S2$ then for any node map ψ satisfying the definition of T-dominance, $\psi^{-1}(A') \ll \psi^{-1}(B')$ in $S1$. •

Depending on the structure of the schema, the above theorem can severely reduce the number of possible node maps that must be considered. Furthermore, the partial order on nodes can be computed in a single traversal of the schema graph.

5.5.2 Edge Maps

Knowledge about constraints on edges can be used to prune the search space of potential edge maps. In general, an edge e of $S1$ can be mapped to any edge g' of $S2$ if surjective functional paths p and r exist in $S1$ between the end points of g' and the end points of e. Furthermore, the existence of constraints on g' implies that certain additional constraints must be present on the paths p and r. If, for example, $g' : C' - D'$ is functional, only injective paths r need to be considered. If g' is a selection edge then only selection paths need be considered. Constraint information alone typically reduces the number of possible maps of an edge to a small number.

6 Applications of Dominance Tests

In the previous section, we presented examples that demonstrated how our characterizations can be used to identify complex correspondences between schemas. A designer may not be able to detect (or keep

[15]In this discussion, *T-dominance* will refer to any of the forms of dominance defined for different sets of transformations considered in this work.

[16]The proof of this requires some inference rules on annotations that are presented elsewhere [13].

track of) a long trail of correspondences between components that might be used to define information preserving mappings. Our characterizations allow this process to be automated. However, the application of our results will be limited if a designer is forced to understand the intricacies of the SIG data model and the correspondence between SIGs and the designer's working model. In this section, we briefly sketch how the results of any computations (that is, transformations or dominance tests) can be easily translated into the terminology of common data models. We present a specific example of how this may be done for the relational model.

Our exposition highlights another important point. Because our methodology is based on local structural manipulations, a dominance detection algorithm may be easily extended to create an interactive tool that aids in the design of equivalent or dominating schemas. Such a tool could be used in defining a view that must express all information in a certain portion of the underlying schema (that is, a view that dominates the underlying schema). Specifically, when a dominance test fails, the algorithm can provide a designer information on how a schema can be changed to obtain the desired relationship. If strict dominance is not required and part of the information in underlying schemas may be hidden, a tool may also aid a designer in verifying that only the desired information is left out of the view.

Consider the relational schemas of Figure 2. In Section 2, we described two scenarios requiring knowledge of the relative information capacity of $R1$ and $R2$. In the second scenario, a designer had changed Schema $R1$ to Schema $R2$. Below, we show how the results of Section 5 (specifically, $\alpha\circ\varsigma$-dominance) can be used by a tool to establish an instance mapping between $R1$ and $R2$.

The SIG equivalents of $R1$ and $R2$ are depicted in Figure 12 (Schemas $S1$ and $S2$, respectively). The test for $\alpha\circ\varsigma$-dominance of these schemas fails. However, since dominance is based on structural properties, we can do more than tell the designer that dominance does not hold. In this example, since all nodes have different types, there is only one possible node mapping ψ between $S1$ and $S2$. If we consider possible edge mappings, we see that l may map to l' and n to n', but there are no surjective functional paths across which the edge g can be mapped to g'. This is the information that needs to be conveyed to the designer. In attempting to establish an instance mapping, a tool may inform the designer of the following: "For Schema R2 to dominate R1 there must be a unique Project associated with every Workstation and every Project must be associated with at least one Workstation."

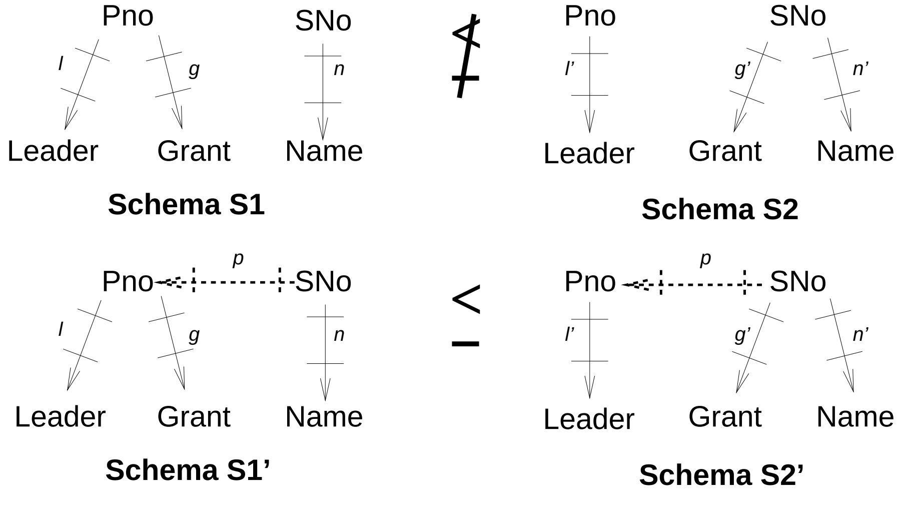


Figure 12: SIG equivalents of the relational schemas of Figure 2.

The above response translates the requirements into commonly understood terms about how entities may be associated. A tool may also be built to translate this information into the terms of the native data model, in this case the relational model. Here, information about how surjective functional paths may be created within the relational model must be used. Functional dependencies can be specified between attributes in the same table or transitively through a chain of attributes in different tables that are also related with inclusion dependencies. In other data models, there may be many ways to define surjective functional relationships. However, a tool can be designed to use knowledge of how such relationships can arise to guide a designer in supplying additional design information.

7 Conclusions

We have shown that no decidable characterization is possible for internal equivalence or dominance on schemas of practical interest. We have given a set of general structural transformations that guarantee internal dominance (or equivalence) and provided complete characterizations of these transformations. While the transformations we have presented are far from exhaustive, they are foundational and represent a solid starting point for the development of dominance tests for more complete sets of transformations. Furthermore, this research methodology should enable practitioners to develop rigorous schema integration methodologies. It permits practitioners to select the structural transformations they consider semantically meaningful and also provides rigorous (sufficiency) tests for both information capacity dominance and equivalence of schemas.

Our methodology enables reasoning about instances of schemas. This reasoning can be applied to schemas developed in any data model. Schema integration is often performed on schemas expressed in semantic data models, which are rich in constructs. When considering information capacity of schemas, the commonality of constructs (in terms of their ability to express instances) must be understood. By reasoning within a data model containing an economy of constructs and constraints, we have been better able to understand the redundancies in constructs of end-user data models. We have also proposed transformations that are, to some extent, "data model independent" in that they use only reasoning about sets and relations.

Many of the equivalence preserving transformations in the literature are due to redundancies in the data model considered and others are specific sequences of structural transformations that we have considered [7, 17, and others]. For example, the *key copying* transformation proposed in a methodology based on the ER+ model is composed of a single $\circ$-transformation followed by an edge ς-transformation to copy a set of key attributes across a functional relationship [17].

We are currently examining other transformations proposed in the literature to understand the basic set theoretic properties they use and develop additional fundamental transformations. We have already developed transformations involving constructed edges and nodes (where a single edge can encode the information of multiple edges) and transformations involving nodes with finite types. We are working on developing characterizations of dominance and equivalence for these additional transformations.

Appendix

A Proof of Lemma 4.2

In this appendix, we prove Lemma 4.2 which we restated here for ease of reference.

Lemma A.1 Let $\Theta(\vec{x})$ and $\Phi(\vec{x})$ be two polynomials in n variables with no constant terms and with coefficients in $\mathcal{N}$, the natural numbers. Then, there exists a SIG schema S, called a *Diophantine encoding* of $\Theta(\vec{x}) = \Phi(\vec{x})$, containing nodes $X_1, X_2, ..., X_n$ such that the equation $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution $\vec{m} = (m_1, m_2, ..., m_n)$, $m_i \in \mathcal{N}$ iff there exists a valid instance $\Im$ for S where $|\Im[X_i]| = m_i$, $1 \leq i \leq n$. •

Proof Let $\vec{x} = (x_1, x_2, ..., x_n)$ and

$$\Theta(\vec{x}) = \sum_{j=1}^{s} a_j x_1^{c_1^j} x_2^{c_2^j} ... x_n^{c_n^j} \qquad \Phi(\vec{x}) = \sum_{j=1}^{t} b_j x_1^{d_1^j} x_2^{d_2^j} ... x_n^{d_n^j}$$

For all j, $1 \leq j \leq s$ and for all k, $1 \leq k \leq n$, $a_j \in \mathcal{N}$ and $c_k^j \in \mathcal{N}$.

For all j, $1 \leq j \leq t$ and for all k, $1 \leq k \leq n$, $b_j \in \mathcal{N}$ and $d_k^j \in \mathcal{N}$.

We define a SIG schema $S = (G, \mathcal{A})$, where $G = (N, E)$, as follows.

For each $x_i, 1 \leq i \leq n$, let $X_i \in N$. For each term, $t_j = a_j x_1^{c_1^j} x_2^{c_2^j} ... x_n^{c_n^j}$, of the polynomial Θ, $(1 \leq j \leq s)$, let Y_j be the following constructed node in N:

$$Y_j = \overbrace{X_1 \times X_1 \times ... \times X_1}^{c_1^j \ times} \times \overbrace{X_2 \times X_2 \times ... \times X_2}^{c_2^j \ times} \times ... \times \overbrace{X_n \times X_n \times ... \times X_n}^{c_n^j \ times}$$

Since Θ contains no constant terms, some $c_k^j \neq 0$, so Y_j is always a valid node in N. For $k = 1$ to a_j, let $Y_j^k \in N$, let $e_j^k : Y_j - Y_j^k \in E$ and let $\mathcal{A}(e_j^k) = \{f, i, s, t\}$. Let the type of each Y_j^k be distinct. Hence, in any valid instance for $\mathcal{A}$ there will be a_j separate simple nodes that must contain sets of the same size. Let Y_j^* and Y be the constructed nodes in N defined below. The polynomial $\Theta(\vec{x})$ is encoded by the node Y.

$$Y_j^* = \sum_{k=1}^{a_j} Y_j^k \qquad Y = \sum_{j=1}^{s} Y_j^*$$

Similarly, the polynomial $\Phi(\vec{x})$ is encoded by a node Z, constructed from the terms of Φ in the same manner as the node Y. Let the edges in this latter construction be label d_j^k, for $1 \leq k \leq b_j$.

The one additional edge $e : Y - Z \in E$ with $\mathcal{A}(e) = \{f, i, s, t\}$ encodes the equality $\Theta(\vec{x}) = \Phi(\vec{x})$.

Let $\vec{m} = (m_1, m_2, ..., m_n)$ where each $m_i \in \mathcal{N}$. We now prove the following: $\vec{m}$ is a solution to the equation $\Theta(\vec{x}) = \Phi(\vec{x})$ iff there exists a valid instance $\Im$ for S such that $|\Im[X_i]| = m_i$, $1 \leq i \leq n$.

$(\Rightarrow)$ Suppose $\Theta(\vec{m}) = \Phi(\vec{m})$. We construct a valid instance $\Im$ of S as follows. Let $\Im[X_i]$ be any set of size m_i, for all i, $1 \leq i \leq n$. Hence, by the definition of SIG instances, for all j, $1 \leq j \leq s$, $|\Im[Y_j]| = (|\Im[X_1]|)^{c_1^j} * (|\Im[X_2]|)^{c_2^j} * ... * (|\Im[X_n]|)^{c_n^j} = m_1^{c_1^j} m_2^{c_2^j} ... m_n^{c_n^j}$. Also, for all j, $1 \leq j \leq t$, $|\Im[Z_j]| = (|\Im[X_1]|)^{d_1^j} * (|\Im[X_2]|)^{d_2^j} * ... * (|\Im[X_n]|)^{d_n^j} = m_1^{d_1^j} m_2^{d_2^j} ... m_n^{d_n^j}$.

For all j, $1 \leq j \leq s$, and k, $1 \leq k \leq a_j$, let $\Im[Y_j^k]$ be a set of size $|\Im[Y_j]|$. Let $\Im[e_j^k]$ be any bijection between sets $\Im[Y_j]$ and $\Im[Y_j^k]$. Clearly, this is a valid instance of the edge e_j^k.

Similarly, for all j, $1 \leq j \leq t$, and k, $1 \leq k \leq b_j$, let $\Im[Z_j^k]$ be a set of size $|\Im[Z_j]|$ and let $\Im[d_j^k]$ be any bijection between the sets $\Im[Z_j]$ and $\Im[Z_j^k]$.

Since the types of the nodes Y_j^k are distinct, the sets $\Im[Y_j^k]$ are disjoint so $|\Im[Y]| = \Theta(\vec{m})$. Similarly, $|\Im[Z]| = \Phi(\vec{m})$. By supposition, $\Theta(\vec{m}) = \Phi(\vec{m})$ so there exists a bijection between the sets $\Im[Y]$ and $\Im[Z]$. We let $\Im[e]$ be any such bijection.

From the definition of $\Im$, it can be seen that $\Im \models \mathcal{A}$ and so is a valid instance of S.

($\Leftarrow$) Conversely, suppose $\Im$ is a valid instance for S. Let $m_i = |\Im[X_i]|$.

Since for all j, $1 \leq j \leq s$, and k, $1 \leq k \leq a_j$, each $\Im[e_j^k]$ is a bijection, by Lemma 4.1, $|\Im[Y_j^k]| = |\Im[Y_j]| = m_1^{c_1^j} m_2^{c_2^j} ... m_n^{c_n^j}$. Also, since the types of the nodes Y_j^k are distinct, the sets $\Im[Y_j^k]$ are disjoint and so $|\Im[Y]| = \Theta(\vec{m})$.

Similarly, for all j, $1 \leq j \leq t$, and k, $1 \leq k \leq b_j$, $|\Im[Z_j^k]| = |\Im[Z_j]|$ and $|\Im[Z]| = \Phi(\vec{m})$.

By Lemma 4.1, since $e : Y - Z$ is a bijection, $|\Im[Y]| = |\Im[Z]|$. Therefore, $\Theta(\vec{m}) = \Phi(\vec{m})$ so $\vec{m}$ is a valid solution to $\Theta(\vec{x}) = \Phi(\vec{x})$. □

B SIG Isomorphism

We present the full definition of SIG isomorphism. We first give preliminary definitions of node and edge maps. An *edge map* is any binary relation on the sets of edges of two schemas that respects inverses.

Definition B.1 A *edge map* between two schemas $S1$ and $S2$ is a binary relation, $\theta : E1 - E2$, such that if $(e, e') \in \theta$ then $(e^\circ, e'^\circ) \in \theta$. •

A *node map* is a binary relation on the sets of nodes of two schemas that respects the product and sum operators on nodes. Constructed nodes may be associated via a node map iff their respective component nodes are associated. Such maps are fully defined by the association between simple nodes of two schemas.

Definition B.2 A *node map* between two schemas $S1$ and $S2$ is a binary relation, $\psi : N1 - N2$, such that:

- for all $(A, A') \in \psi$, $\tau(A) = \tau(A')$;
- for all nodes $A_1 \times A_2 ... \times A_n \in N1$ and $A'_1 \times A'_2 ... \times A'_n \in N2$, $((A_1 \times A_2 ... \times A_n), (A'_1 \times A'_2 ... \times A'_n)) \in \psi$ iff $(A_i, A'_i) \in \psi$ for $1 \leq i \leq n$; and
- for all nodes $A_1 + A_2 ... + A_n \in N1$ and $A'_1 + A'_2 ... + A'_n \in N2$, $((A_1 + A_2 ... + A_n), (A'_1 + A'_2 ... + A'_n)) \in \psi$ iff $(A_i, A'_i) \in \psi$ for $1 \leq i \leq n$. •

We assume that constructed nodes are represented in a normal form that is essentially disjunctive normal form (where $\times$ is "and" and $+$ is "or"). For example, a node $A \times (B + C)$ is represented as the node $A \times B + A \times C$. Furthermore, we assume that differences due to the commutativity and associativity of $\times$ and $+$ are ignored. For example, $A \times B + A \times C$ is the same node as $C \times A + A \times B$. By doing so, we are incorporating the natural equivalence preserving transformations for $+$ and $\times$ constructors. For example, these transformations allow each instance of the node $A \times (B + C)$ to be transformed into a unique instance of the node $C \times A + A \times B$ and vice versa. These transformations are essentially the transformations discussed elsewhere for hierarchical types (without the use of the set constructor) [10]. These equivalences can be proven directly by viewing SIG schemas as algebraic categories. Many of the properties of SIGs that we use in this paper are derived from the categorical structure of SIGs [5].

Definition B.3 Two SIGs are *isomorphic*, denoted $S1 \cong S2$, if there exist a bijective node map $\psi : N1 - N2$ and a bijective edge map $\theta : E1 - E2$ satisfying the following:

- if $e : A - B$ then $\theta(e) : \psi(A) - \psi(B)$ and
- $\mathcal{A}2(\theta(e)) = \mathcal{A}1(e)$ and $\theta(e)$ is a selection (projection) iff e is a selection (projection). •

Certainly, if $S1 \cong S2$ then $S1 \sim_{int} S2$. In general, it may be possible for the annotations on a graph to imply additional annotations. Given a graph G and two SIG schemas $S1 = (G, \mathcal{A}1)$ and $S2 = (G, \mathcal{A}2)$, then $\mathcal{A}1$ logically implies $\mathcal{A}2$ (denoted $\mathcal{A}1 \models \mathcal{A}2$) if every valid instance of $S1$ is a valid instance of $S2$ (that is, $I(S1) \subseteq I(S2)$). Logical implication is certainly a sufficient condition for internal equivalence. We would therefore like to make all possible inferences before computing whether two SIGs are isomorphic. This would permit the detection of internal equivalence for a larger class of schemas. Unfortunately, implication of annotations in SIGs is undecidable. We can however consider the implication of annotations for some set of sound inference rules. In this paper, we assume that the schemas considered are the closures under known inference rules. A discussion of sound inference rules for SIGs and the proof of undecidability of annotation implication are given elsewhere [13].

The SIG formalism may be extended to permit types to be assigned to edges as well as nodes. The definition of SIG isomorphism may be modified to reflect this addition by simply restricting the acceptable edge maps θ to only those maps that preserve edge types.

C Proof of Lemmas 5.6 and 5.7

Before proving Lemma 5.6, we prove that $\circ$-transformations cannot create new surjective functional paths.

Lemma C.1 If $S1 \xrightarrow{\circ}{}^{*} S2$ and there is a surjective functional path $s' : E - F$ in $S2$, then there is some surjective functional path $s : E - F$ in $S1$ with the same constraints. •

Proof The proof is by induction on the length of the sequence of $\circ$-transformations used to transform $S1$ into $S2$.

(Base Case) $S1 \xrightarrow{\{\circ_{g1}^{e1},\ldots\circ_{gn}^{en}\}} S2$. Suppose $s' : E - F$ is a surjective functional path in $S2$. If none of the edges g_i (or their inverses) are contained in s', then s' is a path of $S1$. Suppose some g_i is an edge of s' so $g_i : C - D$ is a surjective function. There must be surjective functional paths $p_i : C - A$, $r_i : D - B$ and an edge $e_i : A - B$ in S in order for the $\circ$-transformation to be valid. Furthermore, since g_i is a surjective function, the edge e_i must be a surjective function and r_i must also be a total injection. The path $s_i = r_i^{\circ} \circ e_i \circ p_i : C - D$ is therefore a surjective functional path in $S1$ and the constraints on s_i are the same as the constraints on g_i. Similarly, if $g_i^{\circ} : D - C$ is an edge of s' then there is a surjective functional path $s_i^{\circ} = p_i^{\circ} \circ e_i^{\circ} \circ r_i : D - C$ in $S1$. For any i, we can therefore replace each occurrence of g_i or g_i° in s' with the paths s_i or s_i°, respectively, to obtain a path $s : E - F$ in $S1$ where the constraints on s and s' are the same.

(Induction Step) Let $S1 \xrightarrow{\circ}{}^{*} T \xrightarrow{\{\circ_{g1}^{e1},\ldots\circ_{gn}^{en}\}} S2$. Applying the same reasoning as for the base case, if there is a surjective functional path in $S2$ then there is a surjective functional path between the same nodes with the same constraints in T. By the induction hypothesis, if there is a surjective functional path in T then there is also some surjective functional path between the same nodes with the same constraints in $S1$. Hence, for any surjective functional path in $S2$ there is some surjective functional path between the same nodes in $S1$ that has the same constraints. □

We now use this result in proving Lemma 5.6 which we restate below.

Lemma C.2 Let $S1 \xrightarrow{\circ}{}^{*} S2$ where $S2$ contains the edge $g' : C - D$. Then, there exist an edge $e : A - B$ and surjective functional paths $p : C - A$, $r : D - B$ (not containing e) in $S1$ such that the constraints on g' in $S2$ are exactly the constraints on the path $r^{\circ} \circ e \circ p$ in $S1$. •

Proof The proof is by induction on the length of the sequence of $\circ$-transformations used to transform $S1$ into $S2$.

(Base Case) $S1 \xrightarrow{\circ} S2$. The lemma is true by the definition of $\circ$-transformations.

(Induction Step) Let $S1 \xrightarrow{\circ}{}^{*} T \xrightarrow{\{\circ_{g'1}^{e'1},\ldots\circ_{g'n}^{e'n}\}} S2$. Each edge of $S2$ that is not one of the g_i' is an edge of T. By the induction hypothesis, the lemma is true for these edges. For each $g_i' : C - D$, there is an edge $e_i' : A - B$ and surjective functional paths $p_i' : C - A$, $r_i' : D - B$ in T. By Lemma C.1, there are surjective functional paths $p_i : C - A$ and $r_i : D - B$ in $S1$ with the same constraints as p_i' and r_i', respectively. If e_i' is an edge of $S1$ then the lemma is proven. Otherwise, by the induction hypothesis, $e_i' : A - B$ was created from an edge $e_i : E - F$ using surjective functional paths $s_i : A - E$ and $t_i : B - F$ in $S1$. So, $e_i : E - F$ is an edge of $S1$ and $s_i \circ p_i : C - E$ and $t_i \circ r_i : D - F$ are surjective functional paths in $S1$ where the constraints on g_i' are exactly the constraints on the path $r_i^{\circ} \circ t_i^{\circ} \circ e_i \circ s_i \circ p_i$. □

We now restate and prove Lemma 5.7.

Lemma C.3 If $S1 \xrightarrow{\alpha\circ}{}^{*} S2$ then there exists an $S1'$ such that $S1 \xrightarrow{\circ}{}^{*} S1' \xrightarrow{\alpha}{}^{*} S2$. •

Proof Suppose $T1 \xrightarrow{\alpha_e} T2 \xrightarrow{O} T3$ where α_e is a specific α-transformation and O is a $\circ$-transformation. Since $T2$ is identical to $T1$ except it has fewer edge constraints or annotations, any $\circ$-transformation that can be applied to $T2$ can be applied to $T1$. So $T1 \xrightarrow{O} T2'$. For each simple $\circ$-transformation in $\circ_{gi}^{ei} \in O$, if e is contained in the path $r_i^\circ \circ e_i \circ p_i$ used by the $\circ$-transformation, let α_{gi} be an α-transformation that removes all the constraints removed by α_e. If e° is contained in the path $r_i^\circ \circ e_i \circ p_i$, then let α_{gi} remove all project and selection constraints and the dual annotations of those removed by α_e. (Totality is the dual of surjectivity and vice versa. Functionality is the dual of injectivity and vice versa.) Let $A = \{\alpha_e\} \cup \{\alpha_{gi} | \circ_{gi}^{ei} \in O\}$ and let $\vec{A}$ be some sequence containing all the transformations in A (we note that α-transformations commute with each other). Then, $T1 \xrightarrow{\circ} T2' \xrightarrow{\vec{A}} T3$.

So, α-transformations can be pushed to the right past $\circ$-transformations. For any sequence of transformations $S1 \xrightarrow{\alpha\circ}{}^{*} S2$, this process can be repeated until all $\circ$-transformations are applied before any α-transformations. To show that this process eventually terminates, we note that no new $\circ$-transformations are created. Additionally, there exist only a finite number of nontrivial α-transformations. So while each step may introduce new α-transformations, by eliminating redundant or trivial α-transformations, the process terminates with a sequence of transformations such that $S1 \xrightarrow{\circ}{}^{*} S1' \xrightarrow{\alpha}{}^{*} S2$. □

D Proof of Theorem 5.13

Before proving Theorem 5.13, we state and prove a few lemmas that will be used in the proof. We find it convenient to define a canonical form for transformations where all isomorphic nodes (nodes connected by bijective selection paths) are removed and then additional transformations applied. Using ς-transformations, we can reintroduce any nodes that remain in the transformed schema $S2$, but removing these nodes first allows us to assume all node ς-transformations are applied to a single node out of a collection of isomorphic nodes.

Definition D.1 The *reduction* of S, denoted $S^{\downarrow}$, is a schema formed by collapsing all isomorphic nodes into a single node. We define a set of equivalence classes of nodes where for any class C_i, and distinct nodes A and B, $A \in C_i$ and $B \in C_i$ iff there exists a bijective selection path $\sigma : A \leftrightarrow\!\!\!\!\!\mid\!\!\mid\!\!\!\!\rightarrow B$. The schema $S^{\downarrow}$ is defined as follows. For each class C_i of S, select a single node $N_i \in C_i$. Let N_i be a node of $N^{\downarrow}$. For each edge $e : A - B$ (that is not a bijective selection edge) where $A \in C_i$ and $B \in C_j$, let $e' : N_i - N_j$ be an edge in $E^{\downarrow}$ and let the constraints of e' be exactly the constraints of e. •

The reduction of a schema S is internally equivalent to S and $S1^{\downarrow} \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ iff $S1 \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$.

Lemma D.1 If $S1^{\downarrow} \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$, then there exist intermediate schemas $T1$ and $T2$ such that $S1^{\downarrow} \xrightarrow{\vec{\Sigma}_1} T1 \xrightarrow{\circ}{}^{*} T2 \xrightarrow{\vec{\Sigma}_2} T3 \xrightarrow{\alpha}{}^{*} S2$ where Σ_1 is a set of node and edge creation ς-transformations and Σ_2 is a set of edge deletion ς-transformations. •

Proof Clearly, all α-transformations can be pushed to the right. So if $S1^{\downarrow} \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$, then $S1^{\downarrow} \xrightarrow{\circ\varsigma}{}^{*} T \xrightarrow{\alpha}{}^{*} S2$.

Suppose the sequence of transformations from $S1^{\downarrow}$ to $S2$ contains a node deletion ς-transformation, ς_A, that removes a node A'. Then, there must be a previous node creation ς-transformation that creates either A or A' (since we begin with the reduction of $S1$ to which no node deletion ς-transformation can be applied). Clearly, we can produce a modified sequence of transformations producing $S2$ and containing no node deletion ς-transformations. The modified sequence contains only the node creation ς-transformations corresponding to nodes that were not later removed by a node deletion ς-transformation in the original sequence.

Let $T1 \xrightarrow{\circ_f^e} T2 \xrightarrow{\varsigma} T3$ where ς is a node creation or an edge creation ς-transformation. These two transformations commute so $T1 \xrightarrow{\varsigma} T2' \xrightarrow{\circ_f^e} T3$.

Let $T1 \xrightarrow{\varsigma} T2 \xrightarrow{\circ_f^e} T3$ where ς is an edge deletion ς-transformation. Any $\circ$-transformation that follows the edge deletion ς-transformation must not use the deleted edge. Hence, the $\circ$-transformation can be applied first. So $T1 \xrightarrow{\circ_f^e} T2' \xrightarrow{\varsigma} T3$.

Let $T1 \xrightarrow{\varsigma_1} T2 \xrightarrow{\varsigma_2} T3$ where ς_1 is an edge deletion ς-transformation and ς_2 is a node creation or an edge creation ς-transformation. Then, $T1 \xrightarrow{\varsigma_2} T2' \xrightarrow{\varsigma_1} T3$.

Hence, node deletion ς-transformations can be removed from the sequence of transformations, node creation and edge creation ς-transformations can be moved left in the sequence and edge deletion ς-transformations can be moved right. So we can create a sequence of transformations $S1^{\downarrow} \xrightarrow{\vec{\Sigma}_1} T1 \xrightarrow{\circ}{}^{*} T2 \xrightarrow{\vec{\Sigma}_2} T3 \xrightarrow{\alpha}{}^{*} S2$ where Σ_1 is a set of node and edge creation ς-transformations and Σ_2 is a set of edge deletion ς-transformations. $\Box$

The next result states that any sequence of transformations can be modified to create a sequence producing the same result schema in which all node creation transformations are applied to nodes in the original schema $S1$ (rather than nodes in intermediate schemas).

Lemma D.2 If $S1^{\downarrow} \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ via any set of transformations then $S1^{\downarrow} \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ via a set of transformations in which all node creation ς-transformations are applied to nodes of $S1^{\downarrow}$. •

Proof By Lemma D.1, we can separate the transformations and produce a set of transformations where all node and edge creation transformations are applied first. Clearly, we can also apply all node creation transformations before edge creation transformations. So $S1^{\downarrow} \xrightarrow{\vec{\Sigma}} T1 \xrightarrow{\vec{X}} S2$ where Σ contains only node creation ς-transformations and X does not contain any node creation ς-transformations.

Suppose there exists a node creation ς-transformation, $\varsigma_{A'}$, that creates a node A'' from a node $A' \notin N1^{\downarrow}$. Then, there must be a bijective selection path in $T1$ from some node A of $S1^{\downarrow}$ to A'. We can therefore replace $\varsigma_{A'}$ with two transformations: a node creation ς-transformation, ς_A, that creates a copy of the node A and a selection edge σ_A and a $\circ$-transformation, $\circ^{\sigma_A}$, that moves σ_A across the bijective path to the node A'. Repeating this process we can obtain a set, Σ', of node creation ς-transformations on nodes of $S1$ such that $S1^{\downarrow} \xrightarrow{\vec{\Sigma}'} T1' \xrightarrow{\circ}{}^{*} T1 \xrightarrow{\vec{X}} S2$. $\Box$

We now prove 5.13 which is restated for convenience.

Theorem D.3 Let $S1$ and $S2$ be SIG schemas. Then, $S1 \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ iff $S1 \preceq_{\alpha\circ\varsigma} S2'$, where $S2' \cong S2$. •

Proof ($\Leftarrow$) We first show that if $S1 \preceq_{\alpha\circ\varsigma} S2'$ then $S1 \xrightarrow{\alpha\circ\varsigma}{}^{*} S2$ for some $S2 \cong S2'$.

1. Let $A \in N1$, where ψ is not defined on A. By Definition 5.8, there is a bijective selection path p in $S1$ from A to some node B where ψ is defined on B. Let O_A be a set of $\circ$-transformations moving all edges incident to A onto the node B. Let ς_A be a node deletion ς-transformation that removes the node A. Let O contain all $\circ$-transformations and Σ_D contain all node deletion ς-transformations created by this process.
2. Let $e \in E1$, where θ is not defined on e. Let ς_e be an edge deletion ς-transformation that removes e. Insert into Σ_D all edge deletion ς-transformations created by this process.
3. Let $A \in N1$, where ψ is defined on A and $\psi(A) = \{A'_0, A'_1, ...A'_n\}$ for some $n \geq 0$. Let $\theta(A) = \{g'_1, g'_2, ..., g'_m\}$ ($m \geq 0$, if $m = 0$ then $\theta(A) = \emptyset$). We define $\imath(A'_0) = A$ and $\imath(A'_i) = A'_i$ for $i \geq 1$. Each node A'_i ($i > 0$) is created by a node creation ς-transformation. If $m < n$ then some of the selection edges created by these ς-transformations have been removed by edge deletion ς-transformations. If $m > n$ then some additional edge creation transformations have been applied.
   - For $i = 1$ to $min(m,n)$, let ς_i be a node creation ς-transformation that creates the node A'_i and selection edge σ_i.
   - For $i = min(m,n) + 1$ to n, let ς_i be a node creation ς-transformation that creates the node A'_i and selection edge σ_i. Also, let ς_{σ_i} be an edge deletion ς-transformation that removes the edge σ_i.
   - For $i = min(m,n) + 1$ to m, let ς_i be an edge creation ς-transformation that creates σ_i.

   Insert into Σ_D the set of all edge deletion ς-transformations and into Σ_C the set of all remaining ς-transformations created in the above process. Let $S1^{\downarrow} \xrightarrow{\vec{\Sigma}_C} T1$.
4. For each $g' : C' - D' \in E2'$, let $\psi^{-1}(C') = C$ and $\psi^{-1}(D') = D$. Since θ is surjective onto the edges of $E2$, one of the following two cases must hold for g'.
   - If $g' \in \theta(A)$ then g' comes from a selection edge $\sigma_{A'} : A' - A$ created by a (node or edge) ς-transformation. Constraints on the original selection edge may have been removed by an α-transformation

and the edge may have been moved (or copied) by a $\circ$-transformation. By Definition 5.8, $\circ_g^\sigma$ is a valid $\circ$-transformation on the schema $T1$ creating an edge $g : C - D$ that may have more constraints than g' (but cannot have fewer constraints). Let $\circ_g^\sigma \in O$ and let $\jmath(g) = g'$.

- If $g' \in \theta(e)$, for some $e : A - B$, then by Definition 5.8, $\circ_g^e$ is a valid $\circ$-transformation. Let $\circ_g^e \in O$ and let $\jmath(g) = g'$.

If any edge g' of $E2'$ has fewer constraints than $\jmath^{-1}(g')$ then, we create an α-transformation to remove any additional constraints. Let A be the set of all α-transformations created.

It is easily verified that $S1 \xrightarrow{\vec{\Sigma}_C} T1 \xrightarrow{\vec{O}} T2 \xrightarrow{\vec{\Sigma}_D} T3 \xrightarrow{\vec{A}} S2$ where $S2 \cong S2'$ via the node bijection $\imath$ and edge bijection $\jmath$.

($\Rightarrow$) Next, we show that if $S1 \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$ then $S1 \preceq_{\alpha\circ\varsigma} S2$ by constructing injections ψ and θ.

Since $S1 \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$ and $S1^\downarrow \overset{\alpha\circ\varsigma}{\longrightarrow}^* S1$, $S1^\downarrow \overset{\alpha\circ\varsigma}{\longrightarrow}^* S2$. By Lemma D.1, there also exists a sequence of transformations of the following form $S1^\downarrow \xrightarrow{\vec{\Sigma}_C} T1 \overset{\circ}{\longrightarrow}^* T2 \xrightarrow{\vec{\Sigma}_D} T3 \overset{\alpha}{\longrightarrow}^* S2$ where Σ_C is a set of node and edge creation ς-transformations and Σ_D is a set of edge deletion ς-transformations. Furthermore, by Lemma D.2, we can assume Σ_C contains only node creation ς-transformations that are applied to nodes in $S1^\downarrow$.

We define the node map ψ and a set of edges E_σ as follows. For each node $A \in N1^\downarrow$, let $A \in \psi(A)$. For each node creation ς-transformation in Σ_C that creates a node A' (from the node A) and a selection edge $\sigma_{A'}$, let $A' \in \psi(A)$ and let $\sigma_{A'} \in E_\sigma$. For each edge creation ς-transformation in Σ_C that creates the edge σ_A, let $\sigma_A \in E_\sigma$.

For each edge g' in $T2$ created from an edge $e \in E1^\downarrow$, we let $g' \in \theta(e)$. For each edge g' in $T2$ created from an edge $\sigma_{A'} \in E_\sigma$, that is not later removed by an transformation in Σ_D, we let $g' \in \theta(A)$. It is easily verified that θ and ψ satisfy Definition 5.8. $\square$

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*, pp. 105–113, Vienna, Austria, Apr. 1993.

[3] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice Hall, New York, NY, 1990.

[4] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, Dec. 1986.

[5] J. L. Bell. *Toposes and Local Set Theories - An Introduction.* Oxford Science Publications, Oxford, U.K., 1988.

[6] M. Davis. Hilbert's Tenth Problem is Unsolvable. *American Mathematical Monthly*, 8(3):233–269, Mar. 1973.

[7] 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*, pp. 25–34, Barcelona, Spain, Sep. 1991.

[8] R. Hull. Relative Information Capacity of Simple Relational Database Schemata. *SIAM Journal of Computing*, 15(3):856–886, Aug. 1986.

[9] R. Hull. A Survey of Theoretical Research on Typed Complex Database Objects. In J. Paredaens, editor, *Databases*, chapter 5, pp. 193–256. Academic Press, London, U.K., 1987.

[10] R. Hull and C. K. Yap. The Format Model: A Theory of Database Organization. *Journal of the ACM*, 31(3):518–537, 1984.

[11] L. A. Kalinichenko. Methods and Tools for Equivalent Data Model Mapping Construction. In *Proc. of the Int'l Conf. on Extending Database Technology*, pp. 92–119, Venice, Italy, Mar. 1990.

[12] 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, Sep. 1992.

[13] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Schema Intension Graphs: A Formal Model for the Study of Schema Equivalence. Technical Report 1185, Dept. of Computer Sciences, U. of Wisconsin, Madison, WI, 1993.

[14] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The Use of Information Capacity in Schema Integration and Translation. In *Proc. of the Int'l Conf. on Very Large Data Bases*, pp. 120–133, Dublin, Ireland, Aug. 1993.

[15] C. Ó'Dúnlaing and C. K. Yap. Generic Transformation of Data Structures. In *Sym. on Foundations of Computer Science*, pp. 186–195, Chicago, IL, Nov. 1982.

[16] A. Rosenthal and D. Reiner. Theoretically Sound Transformations for Practical Database Design. In *Proc. of the Int'l Conf. on Entity-Relationship Approach*, pp. 115–131, New York, NY, Nov. 1987.

[17] A. Rosenthal and D. Reiner. Tools and Transformations - Rigorous and Otherwise - For Practical Database Design. Technical report, MITRE Corp., Feb. 1993.

[18] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[19] A. P. Sheth and H. Marcus. Schema Analysis and Integration: Methodology, Techniques, and Prototype Toolkit. Technical Report TM-STS-019981/1, Bellcore, Mar. 1992.

[20] P. Shoval and S. Zohn. Binary-Relationship Integration Methodology. *Data and Knowledge Engineering*, 6:225–250, 1991.

[21] M. Templeton, H. Henley, E. Maros, and D. J. Van Buer. InterViso: Dealing with the Complexity of Federated Database Access. Technical report, Data Integration Inc., Los Angeles, CA, Dec. 1992.

[22] G. Thomas, G. R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous Distributed Database Systems for Production Use. *ACM Computing Surveys*, 22(3):237–266, Sep. 1990.