Schema Intension Graphs: A Formal Model for the Study of Schema Equivalence

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

Technical Report #1185

January 1994

Technical Report 1185

Schema Intension Graphs: A Formal Model for the Study of Schema Equivalence

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

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

Abstract

We develop a formal data model, the *Schema Intension Graph* (SIG) model, to aid in understanding the relative information capacity of schemas containing constraints. The basic building blocks of the SIG model are sets that may be combined by the nested application of union and product constructors. The model also permits the expression of binary relations on sets and simple integrity constraints on these relations. We discuss the motivation used in designing the model and establish some fundamental results on the model. We consider the problem of constraint implication in the SIG model and give a sound and complete set of implication rules for a subclass of SIG schemas, called simple SIG schemas. The general constraint implication problem is shown to be undecidable. Finally, we consider information capacity preserving translations of a subclass of relational schemas with functional and inclusion dependencies into simple SIG schemas. These translations assist in determining the relative information capacity of relational schemas.

*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. Much of the existing work on schema translation and integration relies on an intuitive, rather than formal, notion of correctness. As a result, solutions are motivated by the needs of specific classes of examples and often do not generalize.

Our previous work has highlighted the need for a formal notion of equivalence [MIR93a]. Specifically, we examined the notion of *relative information capacity* [Hul86] and identified anomalies that can arise when using transformations that do not guarantee that information capacity is preserved. Some recent work on schema translation has successfully used information capacity equivalence as a basis for judging the correctness of translation algorithms [MS92, RR87]. However, there are only limited results on testing for information capacity equivalence and dominance of arbitrary schemas. To be usable in a practical context, a characterization of both information capacity equivalence and dominance of schemas is needed.

In this paper, we develop a formal data model, the *Schema Intension Graph* (SIG) model, that is designed to aid in developing these formal results. We have defined 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. 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 this, 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 [Hul86] or models based on complex types [AH88, Hul87, HY84, OY82]. Instead of extending complex types with constraints, we 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 in Section 3. 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 [RR93]. Specifically, the constraints we include in the model are local (that is, they are robust to schema 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.

In Section 2, we define information capacity dominance and equivalence. In Section 3, we briefly explore existing results on information capacity equivalence and the additional results we wish to obtain. We discuss how this task motivates the design of the SIG data model. In Section 4, we define the SIG data model. In Section 5, we present foundational results necessary to reason about the equivalence of SIG schemas. We define SIG isomorphism, give a sound and complete set of annotation implication rules for a class of SIG schemas and prove that annotation implication in general is undecidable. Finally, in Section 6, we briefly illustrate the use of the SIG model by presenting an algorithm that translates an important class of relational schemas into SIG schemas.

2 Information Capacity

The measure of correctness for schema transformations that we use is based on the preservation of the information capacity of schemas.[1] The *information capacity* of a schema S is measured by the set of all valid instances of S. Let $I(S)$ denote the set of all valid instances of schema S. A schema conveys information about the universe it models. This information is essentially captured by $I(S)$ which contains all the possible states of the modeled universe.

An instance $\Im \in I(S)$ of a schema S uses some (finite) subset of symbols drawn from some universe of symbols U. We denote this subset $Sym(\Im)$. In addition to $I(S)$, we consider $I_Y(S)$, the set of all instances of S that contain only symbols in the set $Y \subseteq U$, $I_Y(S) = \{\Im \mid \Im \in I(S) \text{ and } Sym(\Im) \subseteq Y\}$.

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. This notion is formalized using the existence of invertible (that is, injective) functions between the sets of instances of two schemas.

Definition 2.1 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 2.2 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 $f : I_Y(S1) \rightarrow I_Y(S2)$. •

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 information capacity equivalence may be built.

Definition 2.3 The schema $S2$ *dominates* $S1$ *absolutely*, denoted $S1 \preceq_{abs} S2$, if there is a finite $Z \subseteq U$ such that for each (finite) $Y \supseteq Z$ there exists an information preserving mapping $f : I_Y(S1) \rightarrow I_Y(S2)$ (or equivalently, $|I_Y(S1)| \leq |I_Y(S2)|$).[2] •

Definition 2.4 Let $S1$ and $S2$ be schemas. Also, $S1$ and $S2$ are *absolutely equivalent*, denoted $S1 \sim_{abs} S2$, if there is a finite $Z \subseteq U$ such that for each $Y \supseteq Z$ there exists an equivalence preserving mapping $f : I_Y(S1) \rightarrow I_Y(S2)$ (or equivalently $|I_Y(S1)| = |I_Y(S2)|$). •

[1] Relative information capacity was first studied by Hull [Hul86]. Information capacity has also been applied to a number of translation and integration problems [AH88, HY84, MS92, OY82, RR87, and others].

[2] Dominance is most interesting when the sets $I_Y(S1)$ and $I_Y(S2)$ are finite. For schemas in most common data models (including the SIG model), the set $I_Y(Si)$ is finite if Y is finite.

3 Motivation for the SIG Model

Studies of absolute equivalence have focused on schemas without constraints. For the relational model without dependencies, two schemas are equivalent iff they are identical (up to renaming of attributes and relations) [Hul86]. Additional work has considered the relative information capacity of complex types formed by the recursive application of product, set or union constructors on both infinite and finite base types. For complex types containing only infinite base types, absolute equivalence can be characterized by a set of natural restructuring operators [HY84]. The restructuring operators are used to define a normal form for these schemas such that two schemas are absolutely equivalent if and only if their normal forms are isomorphic. This result has been generalized to schemas that include finite as well as infinite base types [AH88]. Again, a (decidable) characterization of absolute equivalence for the extended set of schemas is given that is based on a set of restructuring operators. Characterizations of absolute dominance for complex types are not known.

Schemas that arise in practice contain constraints that define which instances of a schema are meaningful in a certain context. To make use of reasoning about information capacity in schema translation and integration tools, we must therefore extend the above results to consider algorithms for deciding equivalence of schemas with constraints. Our primary motivation in defining the SIG data model is to have a convenient formalism in which we can develop such results. Given this goal we established two requirements for the SIG model. First, the model must be data-centric rather than type-centric. Second, any reasoning about schema equivalence must be accessible and done in a form that is easily conveyed back to a schema designer. These requirements are discussed below.

3.1 Data-centricity

By attaching a set of type definitions to an entity set, one can express constraints on the structure of individual entities. For example, in many complex type (or object) models constraints are placed on the internal structure of entities [Hul87]. Similarly, within the relational model, data dependencies express constraints on the structure of individual entities (where the entities are subsets of n-way products of sets). In such models, the interpretation of constraints are tied closely to the types or structures over which they are expressed. 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. We therefore choose not to use any of these existing models and instead defined a model that is inherently "data-centric" as opposed to "type-centric".

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** $\times$ **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** $\times$ **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 (whose instances may lie in some range of permissible values). Rather, a constructor defines a unique new set from the instances of the types on which it operates. 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 $\times$ is viewed as a type constructor!) We have defined 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.

3.2 Accessibility

We wish to develop results (algorithms for testing information capacity equivalence and dominance) that are directly usable in schema integration and translation tools. Achieving this goal will be easier if the formalism in which the results are couched is accessible to schema designers. We therefore 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 [RR93]. Specifically, the constraints we include in the model are local (that is, they are robust to schema 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. These practical requirements also led us to design a model that is not subject to assumptions such as the relationship uniqueness assumption [AP82] that requires there be only one relationship expressed between any two sets.

Additionally, to simplify our task, we include in the model only a minimum set of constructs and constraints necessary to model a large class of commonly occurring schemas. They also appear in some form in most common data models, and are therefore widely recognized as being useful.

4 Model Definition

The basic building blocks of the SIG 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. The model also permits the expression of binary relations between pairs of sets and simple constraints 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]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 [BW90].

4.1 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.

4.1.1 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, $\Im[\Pi_A^B] = \{((b,c),b) \mid (b,c) \in \Im[A] \text{ and } b \in \Im[B]\}$.

4.1.2 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 S_A and S_B be two sets. A binary relation $r : S_A - S_B$ is *total* (denoted $e : S_A \vdash\!\!- S_B$) if it is defined for all elements of S_A; *surjective* ($e : S_A -\!\!\dashv S_B$) if it is defined for all elements of S_B; *functional* ($e : S_A \longrightarrow S_B$) if an element of S_A determines at most one element of S_B; and *injective* ($e : S_A \longleftarrow S_B$) if an element of S_B determines at most one element of S_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 S_A and S_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 $e \in E$, $\mathcal{A}(e) \subseteq \{f, i, s, t\}$. For every edge e, $f \in \mathcal{A}(e)$ iff $i \in \mathcal{A}(e^o)$ and $s \in \mathcal{A}(e)$ iff $t \in \mathcal{A}(e^o)$. 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 $Gi = (Ni, Ei)$. For two annotation functions $\mathcal{A}1$ and $\mathcal{A}2$ on a graph G, we say that $\mathcal{A}1 \subseteq \mathcal{A}2$ (respectively, $\mathcal{A}1 = \mathcal{A}2$) if for every edge e of G, $\mathcal{A}1(e) \subseteq \mathcal{A}2(e)$ (respectively, $\mathcal{A}1(e) = \mathcal{A}2(e)$).

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

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 $r : S_A - S_B$ is a binary relation then for $a \in S_A$, $r(a)$ denotes the set of elements of S_B associated with the element a in r. Additionally, if $s : S_B - S_C$ is a binary relation then $s \circ r : S_A - S_C$ 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 4.1 Let $G = (N, E)$ be a SIG and $\mathcal{A}$ an annotation function on G. A *path*, $p : A_1 - A_k$, in G is a (possibly empty) sequence of edges $e_1 : A_1 - A_2$, $e_2 : A_2 - A_3$, ..., $e_{k-1} : A_{k-1} - A_k$ and is denoted $e_{k-1} \circ e_{k-2} \circ \ldots \circ e_1$. 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.

4.2 An Example SIG

Figure 1 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. 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.

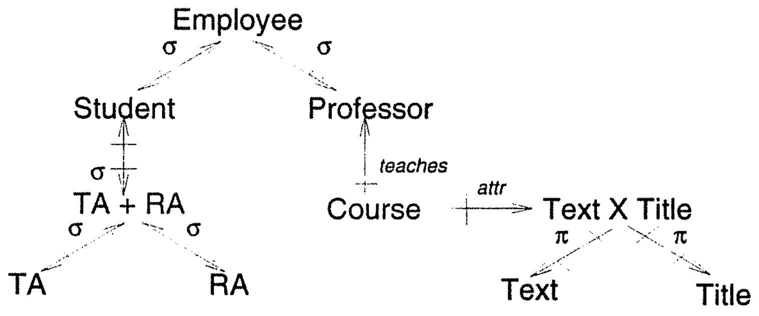

Figure 1: An example SIG schema.

5 Reasoning about SIGs

We consider how basic correspondences between SIG schemas may be established. Our ultimate goal is to understand the relative information capacity of SIG schemas and so we present correspondences that are sufficient conditions for absolute equivalence. Under information capacity equivalence, differences in schemas due to naming of constructs are irrelevant. Hence, in any 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. We then consider logical implication of annotations on SIG schemas.

[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.

5.1 SIG Isomorphism

SIG isomorphism is a special case of graph isomorphism that is 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). Before presenting the 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 5.1 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 if and only if their respective component nodes are associated. Such maps are fully defined by the association between simple nodes of two schemas.

Definition 5.2 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 $1 \leq i \leq n$. •

Definition 5.3 Two SIG schemas 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 $S1 \sim_{abs} S2$. Furthermore, we can compute when $S1 \cong S2$ using a standard graph isomorphism algorithm that is modified to take into account construction and types of nodes.

In what follows, 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) [HY84].

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 to only those maps that preserve edge types.

5.2 Annotation Implication

Consider a graph G and two SIG schemas $S1 = (G, \mathcal{A}1)$ and $S2 = (G, \mathcal{A}2)$. From the definition of valid instances, it is easily verified that if $\mathcal{A}1 \subseteq \mathcal{A}2$, then any valid instance of $S2$ is necessarily a valid instance of $S1$, that is, $I(S2) \subseteq I(S1)$. Furthermore, a set of annotations may imply additional annotations. Hence, it may be the case that for distinct $\mathcal{A}1$ and $\mathcal{A}2$, $I(S1) = I(S2)$. We say that $\mathcal{A}1$ *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 absolute dominance. We would therefore like to make all possible inferences before computing whether two SIG schemas are isomorphic. This would permit the detection of absolute equivalence for a larger class of schemas. To understand implication in SIG schemas, we note that annotations allow the expression of various cardinality constraints between sets.

Lemma 5.1 Let S_A and S_B be two finite sets. There exists a surjective function from S_A to S_B iff $|S_A| \geq |S_B|$. •

Proof ($\Rightarrow$) Let $f : S_A - S_B$ be a surjective function. Under f, all elements of S_B are paired with some element of S_A and no element of S_A is paired with more than one element of S_B. Hence, there must be at least as many elements in S_A as in S_B.

($\Leftarrow$) Let $S_{A'}$ be a subset of S_A such that $|S_{A'}| = |S_B|$. Any two finite sets of equal size can be put into bijective correspondence. Let $f : S_{A'} - S_B$ be any such bijection. Then, f is a surjective function from S_A to S_B. □

Corollary 5.2 Let $S = (G, \mathcal{A})$ be a SIG schema where $G = (N, E)$. Let $e : A - B \in E$. If $\mathcal{A}(e) \supseteq \{f, s\}$ then for all $\Im \in I(S)$, $|\Im[A]| \geq |\Im[B]|$. If $\mathcal{A}(e) \supseteq \{i, t\}$ then for all $\Im \in I(S)$, $|\Im[A]| \leq |\Im[B]|$. •

Proof The corollary is an immediate consequence of Lemma 5.1 and the definition of SIG instances. □

Using cardinality arguments, we can derive inference rules on annotations. The next lemma states that if there exists a surjective functional path from a node to itself then every edge on the path is a bijection.

Theorem 5.3 Let $S1 = (G, \mathcal{A}1)$ and $S2 = (G, \mathcal{A}2)$ be two SIG schemas where $G = (N, E)$. Let $p : A_1 - A_1$ be a path from a node to itself where $e_1 : A_1 - A_2$, $e_2 : A_2 - A_3$, ..., $e_k : A_k - A_1$ and $p = e_k \circ e_{k-1} \circ ... \circ e_1$. Let $\mathcal{A}1(e_i) \supseteq \{f, s\}$ for all i, $1 \leq i \leq k$ (or $\mathcal{A}1(e_i) \supseteq \{i, t\}$ for all i, $1 \leq i \leq k$). Let $\mathcal{A}2(e_i) = \{f, i, s, t\}$ for all i, $1 \leq i \leq k$ and $\mathcal{A}2(e) = \emptyset$ for all e not in p. Then, $\mathcal{A}1 \models \mathcal{A}2$. •

Proof Let $\mathcal{A}1(e_i) \supseteq \{f, s\}$ for all i, $1 \leq i \leq k$. Let $\Im \in I(S1)$ be any instance of $S1$. By Corollary 5.2, $|\Im[A_1]| \leq |\Im[A_2]| \leq ... \leq |\Im[A_k]| \leq |\Im[A_1]|$. Therefore, $|\Im[A_1]| = |\Im[A_2]| = ... = |\Im[A_k]|$. Any surjective function between two sets of the same size is necessarily a bijection. Hence, $\Im[e_i]$ is a bijection for each i, $1 \leq i \leq k$. Similarly, if $\mathcal{A}1(e_i) \supseteq \{i, t\}$ for all i, $1 \leq i \leq k$ then $\Im[e_i]$ is a bijection for each i, $1 \leq i \leq k$.

Therefore, for any $\Im \in I(S1)$, $\Im$ is also an instance of $S2$ and so $\mathcal{A}1 \models \mathcal{A}2$. □

To be useful in a practical setting, we need a decision procedure for determining when $\mathcal{A}1 \models \mathcal{A}2$. In particular, we would like an algorithm for computing an annotation function containing all annotations logically implied by a given annotation function. Such an annotation function is called the *closure*.

Definition 5.4 Let $S = (G, \mathcal{A})$ be a SIG schema. The *closure* of $\mathcal{A}$ is an annotation function $\mathcal{A}^*$ on G such that for all annotation functions $\mathcal{B}$, if $\mathcal{A}^* \models \mathcal{B}$ then $\mathcal{B} \subseteq \mathcal{A}^*$. •

5.3 Decidability of Annotation Implication for Simple SIG Schemas

Before considering the general problem of annotation implication in SIG schemas, we consider implication in a subclass called *simple* SIG schemas.

Definition 5.5 A *simple SIG schema* is a SIG schema $S = (G, \mathcal{A})$ where $G = (N, E)$ and N contains only simple nodes (that is, no cross product or sum nodes). •

Below, we give a sound and complete set of axioms for annotation implication in simple schemas. By sound, we mean that if an annotation is implied by the rules, then the annotation is in fact in $\mathcal{A}^*$. By complete, we mean that using the implication rules we can infer all annotations in $\mathcal{A}^*$.

Theorem 5.3 gives one inference rule. To show that no other (nontrivial) inference rules exist, we will construct an instance of a SIG schema that satisfies the annotations implied by this theorem and no other annotations. The construction uses the following lemma that proves the existence of binary relations that satisfy specified subsets of annotations and violate all other annotations.[5]

Lemma 5.4 Let S_A and S_B be finite sets where $|S_A| > 2$ and $|S_B| > 2$ and $r : S_A - S_B$ a binary relation defined on them. Let $\alpha \subseteq \{f, i, s, t\}$, and let $\bar{\alpha} = \{f, i, s, t\} - \alpha$.

1. If $|S_A| < |S_B|$ then $\forall\, \alpha \not\supseteq \{f, s\}$, $\exists\, r$ that satisfies α and does not satisfy $\bar{\alpha}$.
2. If $|S_A| > |S_B|$ then $\forall\, \alpha \not\supseteq \{i, t\}$, $\exists\, r$ that satisfies α and does not satisfy $\bar{\alpha}$.
3. If $|S_A| = |S_B|$ then $\forall\, \alpha$ where $\alpha \not\supseteq \{f, s\}$ and $\alpha \not\supseteq \{i, t\}$, $\exists\, r$ that satisfies α and does not satisfy $\bar{\alpha}$. •

Proof The proof is by a straightforward case analysis. □

Theorem 5.5 The following set of inference rules is sound and complete for implication of annotations in simple SIG schemas.

R1 *Trivial Inference:* For all annotations $\mathcal{A}$, $\mathcal{A}$ implies $\mathcal{A}$.

R2 *Bijective Inference:* Let $S1 = (G, \mathcal{A}1)$ and $S2 = (G, \mathcal{A}2)$ be two SIG schemas and let $G = (N, E)$. Let $p : A_1 - A_1$ be a path from a node to itself where $e1 : A_1 - A_2$, $e2 : A_2 - A_3$, ..., $ek : A_k - A_1$ and

[5]We take some liberties with our notation for convenience. A binary relation is said to satisfy an annotation f (respectively, $i, s,$ or t) if it is functional (respectively, injective, surjective or total).

$p = e_k \circ e_{k-1} \circ \ldots \circ e_1$. Let $\mathcal{A}1(e_i) \supseteq \{f, s\}$ for all i, $1 \leq i \leq k$ (or $\mathcal{A}1(e_i) \supseteq \{i, t\}$ for all i, $1 \leq i \leq k$). Let $\mathcal{A}2(e_i) = \{f, i, s, t\}$ for all i, $1 \leq i \leq k$ and $\mathcal{A}2(e) = \emptyset$ for all e not in p. Then, $\mathcal{A}1$ implies $\mathcal{A}2$. •

Proof Rule R1 is clearly sound and the soundness of rule R2 over all SIG schemas was proven in Theorem 5.3. We now prove completeness.

Let $S = (G, \mathcal{A})$ be a simple SIG where $G = (N, E)$. Let $S = (G, \mathcal{A}1)$ where $\mathcal{A}1$ contains all annotations that can be inferred from $\mathcal{A}$ using the stated inference rules. Finally, let $\mathcal{A}2 \supset \mathcal{A}1$.

We claim that $\mathcal{A}1 \not\models \mathcal{A}2$. To show this, we will construct an instance $\Im$ of G such that $\Im \models \mathcal{A}1$ and $\Im \not\models \mathcal{A}2$.

We can partition the nodes of $S1$ into disjoint subsets of nodes, called *bijectively connected components* (BCCs), where A and B are in the same BCC if and only if there is a bijective path from A to B (in $S1$). We define a partial order on the set of all BCCs, $<_B$, where $\beta_1 <_B \beta_2$ if and only if $\beta_1 = \beta_2$ or there exist nodes $A \in \beta_1$ and $B \in \beta_2$ and a total injective path from A to B (or equivalently, a surjective functional path from B to A.) By Inference Rule R2, $<_B$ is a partial order.

Now $\Im$ can be constructed as follows:

- *Nodes:* The assignment of sets to nodes may be made in any way subject to the following constraints.
  - If β is a BCC, then for all $A \in \beta$, let $|\Im[A]| = n(\beta)$ where $n(\beta)$ is an integer greater than 2.
  - If $\beta_1 <_B \beta_2$ and $\beta_1 \neq \beta_2$, then $n(\beta_1) < n(\beta_2)$.

  Such an assignment is possible since $<_B$ is a partial order.

- *Edges:* For each edge $e : A - B$, $\Im[e]$ is any relation constructed as follows.
  - *An edge between nodes in the same BCC.* Let A and B be in the same BCC so $|\Im[A]| = |\Im[B]|$ and there is some bijective path p from B to A. If $\mathcal{A}1(e) \supseteq \{f, s\}$ then $p \circ e : A - A$ is a surjective functional path from A to itself. Hence, by inference rule R2, $\mathcal{A}1(e) = \{f, i, s, t\}$. Similarly, if $\mathcal{A}1(e) \supseteq \{i, t\}$ then $p^\circ \circ e^\circ : B - B$ is a surjective functional path from B to itself. So, again by inference rule R2, $\mathcal{A}1(e) = \{f, i, s, t\}$. If $\mathcal{A}1(e) = \{f, i, s, t\}$ then let $\Im[e]$ be any bijection between $\Im[A]$ and $\Im[B]$. Otherwise, $\mathcal{A}1(e) \not\supseteq \{s, f\}$ and $\mathcal{A}1(e) \not\supseteq \{i, t\}$ so by Lemma 5.4-3, there exists a binary relation that satisfies exactly the annotations of $\mathcal{A}1(e)$. Let $\Im(e)$ be such a binary relation.
  - *An edge between comparable nodes.* Let $A \in \beta_1$ and $B \in \beta_2$ where $\beta_1 \neq \beta_2$.

    If $\beta_1 <_B \beta_2$ then $\Im[A] < \Im[B]$ and by the definition of $<_B$ there is some total injective path $p : A - B$ in $S1$. If $\mathcal{A}1(e) \supseteq \{f, s\}$ then $p^\circ \circ e : A - A$ is a surjective functional path from the node A to itself, so by the inference rule R2, $\mathcal{A}1(e) = \{f, i, s, t\}$. However, this contradicts the assumption that A and B are in different BCCs. Hence, $\mathcal{A}1(e) \not\supseteq \{f, s\}$, so by Lemma 5.4-1, there is a binary relation on the sets $\Im[A]$ and $\Im[B]$ that satisfies exactly the annotations on e. Let $\Im(e)$ be such a binary relation.

If $\beta_2 <_B \beta_1$ then $\Im[A] > \Im[B]$ and by the definition of $<_B$ there is some total injective path $p : B - A$. If $\mathcal{A}1(e) \supseteq \{i,t\}$ then $e^\circ \circ p^\circ : A - A$ is a surjective functional path from the node A to itself, so by the inference rule R2, $\mathcal{A}1(e) = \{f,i,s,t\}$. However, this contradicts the assumption that A and B are in different BCCs. Hence, $\mathcal{A}1(e) \not\supseteq \{i,t\}$, so by Lemma 5.4-2, there is a binary that satisfies exactly the annotations on e. Let $\Im(e)$ be such a binary relation.

- *An edge between incommensurate BCC.* Let $A \in \beta_1$ and $B \in \beta_2$ where $\beta_1 \neq \beta_2$, $\beta_1 \not<_B \beta_2$ and $\beta_2 \not<_B \beta_1$. Since $\beta_1 \not<_B \beta_2$, $\mathcal{A}1(e) \not\supseteq \{f,s\}$. Similarly, since $\beta_2 \not<_B \beta_1$, $\mathcal{A}1(e) \not\supseteq \{i,t\}$. Hence, by Lemma 5.4, (and regardless of the respective sizes of $\Im(A)$ and $\Im(B)$), there is a binary relation that satisfies exactly the annotations on e. Let $\Im(e)$ be such a binary relation.

By the construction of $\Im$, $\Im \models \mathcal{A}1$. Furthermore, $\Im$ models exactly those annotations in $\mathcal{A}1$, so $\Im \not\models \mathcal{A}2$. Since this is true for any $\mathcal{A}2$ containing strictly more annotations than $\mathcal{A}1$, $\mathcal{A}1$ contains all annotations logically implied by $\mathcal{A}$. Hence, the closure $\mathcal{A}^*$ contains exactly the annotations implied by Rules R1 and R2. □

We briefly sketch how the closure can be efficiently computed. We can construct a graph H from a schema S where the nodes of H are the nodes of S and the edges of H are the total injective edges of S. We can partition the nodes of H into strongly connected components such that two nodes, A and B, are in the same component if and only if there is at least one path from A to B and from B to A. Every surjective functional (or total injective) edge between nodes within a strongly connected component corresponds to a bijective edge in $\mathcal{A}^*$. Furthermore, there are no other annotations in $\mathcal{A}^*$ that are not in $\mathcal{A}$. The strongly connected components of a graph can be computed in time linear in the size of the graph [Tar72]. Using a simple modification to this result, the closure under annotation implication of a simple SIG schema can be computed in time linear in the size of the schema.

5.4 Undecidability of Annotation Implication for SIG schemas

We now consider the implication problem for the class of all SIG schemas. We will show that this problem is undecidable. 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 ranging over the natural numbers (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 [Dav73].

The next lemma states that Diophantine equations without constant terms may be "encoded" by a SIG schema. Specifically, given a Diophantine equation $\Theta(\vec{x}) = \Phi(\vec{x})$, we construct a schema S such that every valid instance of S corresponds to a solution to the Diophantine equation and vice versa. Our reduction will then consider an arbitrary Diophantine equation (possibly with constant terms) $\Theta(\vec{x}) = \Phi(\vec{x})$ and construct a SIG schema corresponding to the equation $\Theta(\vec{x})w = \Phi(\vec{x})w$.

Lemma 5.6 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 $\mathcal{A}(e_j^k) = \{f, i, s, t\}$. 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 the type of each Y_j^k be distinct. 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 Corollary 5.2, $|\Im[Y_j^k]| = |\Im[Y_j]| = m_1^{c_1^j} m_2^{c_2^j} ... m_n^{c_n^j}$. 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 Corollary 5.2, 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 $\Theta(\vec{x}) = \Phi(\vec{x})$. □

Solutions to the logical implication problem for annotations over SIG schemas together with the above encoding can be used to determine when a Diophantine equation has a solution in the natural numbers. From this, we can conclude that implication in SIG schemas is undecidable.

Theorem 5.7 The annotation implication problem for SIG schemas is undecidable. •

Proof We will construct a graph G and annotation functions $\mathcal{B}1$ and $\mathcal{B}2$ for which the logical implication problem is undecidable and conclude that the implication problem for arbitrary SIG schemas is undecidable.

Let $\Theta(\vec{x})$ and $\Phi(\vec{x})$ be two arbitrary 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 $\mathcal{A}1$ be an annotation function on G' where $\mathcal{A}1(e) = \mathcal{A}(e)$ for all $e \in E$ and $\mathcal{A}1(d) = \emptyset$. Let $\mathcal{A}2$ be the annotation function on G' where $\mathcal{A}2(e) = \mathcal{A}(e)$ for all $e \in E$ and $\mathcal{A}2(d) = \{t\}$. Let $S1$ and $S2$ be the two SIG schemas $(G', \mathcal{A}1)$ and $(G', \mathcal{A}2)$ respectively.

By the definition of instances, every instance of $S2$ is an instance of $S1$, so $I(S2) \subseteq I(S1)$. Also, by the definition of implication, $\mathcal{A}1 \models \mathcal{A}2$ iff $I(S1) \subseteq I(S2)$.

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 $\Im$ of $S1$ or $S2$ restricted to the nodes and edge of S is a valid instance of S.

We now prove that $\mathcal{A}1 \models \mathcal{A}2$ iff $\Theta(\vec{x}) = \Phi(\vec{x})$ has no solution.

($\Rightarrow$) Suppose $\mathcal{A}1 \models \mathcal{A}2$, so $I(S1) \subseteq I(S2)$. Suppose $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution $\vec{m}$. Then, $\Theta(\vec{x})w = \Phi(\vec{x})w$ has solutions for which $w \neq 0$. By Lemma 5.6, 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 $\Im[W]$). Therefore $I(S1) \not\subseteq I(S2)$ which contradicts the assumption that $\mathcal{A}1 \models \mathcal{A}2$. So the equation $\Theta(\vec{x}) = \Phi(\vec{x})$ has no solutions.

($\Leftarrow$) Now suppose $\mathcal{A}1 \not\models \mathcal{A}2$. Then, there exists some instance $\Im$ such that $\Im \in I(S1)$ and $\Im \notin I(S2)$. The instance $\Im$ must populate the edge 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 5.6, $\Theta(\vec{x})w = \Phi(\vec{x})w$ has a solution for which $w \neq 0$. Therefore, $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution. So, if $\Theta(\vec{x}) = \Phi(\vec{x})$ has no solution then $\mathcal{A}1 \models \mathcal{A}2$.

Since the problem of determining whether $\Theta(\vec{x}) = \Phi(\vec{x})$ has a solution is undecidable, annotation implication of SIG schemas is also undecidable. □

6 Equivalence Preserving Translations

We turn to the question of equivalence preserving translations of relational schemas into SIG schemas. Such translations will allow us to use results on the relative information capacity of SIG schemas to understand the relative information capacity of relational schemas.

6.1 Relational Schemas

We first present a formal definition of relational schemas.

Let Δ be a set of attributes and $\mathcal{T}$ a set of type. For each $A_i \in \Delta$, $\tau(A_i) = T_i \in \mathcal{T}$ is the type of A_i.

A relational table R is a list of attributes and their associated types, $R = [A_1 : T_1, A_2 : T_2, ..., A_n : T_n]$. (When convenient, the types may be omitted and R may be denoted as a list of attributes.)

For an instance $\Im[R]$ of a relational table R, $\Im[R] \subseteq T_1 \times T_2 \times ... \times T_n$. An element $(a_1, ..., a_n) \in \Im[R]$ is called a *tuple.* If $S = [A_{i_1}, ..., A_{i_s}]$ where for all j, $1 \leq j \leq s$, A_{i_j} is an attribute of R. Then, $\Im[S]$ is the projection of $\Im[R]$ onto the attributes of S: $\Im[S] = \{(a_{i_1}, ...a_{i_s}) \mid \exists \text{ some } (a_1, a_2, ..., a_n) \in \Im[R]\}$.

We consider two types of constraints on the set of valid instances of relational tables.

- **Functional Dependencies.** Let X and Y be lists of attributes of R. The *functional dependency* $X \rightarrow Y$ holds on an instance $\Im[R]$ if all tuples of $\Im[R]$ that agree on each of the attributes in X also agree on the attributes in Y. If $X \rightarrow R$ and for no $Y \subset X$, $Y \rightarrow R$, then X is called a *key* and $X \rightarrow R$ is a key dependency. A key is *simple* if it contains a single attribute.
- **Inclusion Dependencies.** Let X be a list of attributes of R and Y be a list of attributes of S of the same size as X. The *inclusion dependency* $R[X] \subseteq S[Y]$ holds on instances $\Im[R]$ and $\Im[S]$ if $\Im[X] \subseteq \Im[Y]$. An inclusion dependency is *unary* if X (and therefore Y) contains a single attribute. A pair of inclusion dependencies $R[X] \subseteq S[Y]$ and $S[Y] \subseteq R[X]$ is called a *bidirectional* inclusion dependency and denoted $R[X] = S[Y]$.

A relational schema $\mathcal{R}$ is a set of relational tables (each with distinct attributes) and a set of dependencies. An instance of a schema $\mathcal{R}$ is a set of instances of the relational tables in $\mathcal{R}$ that collectively satisfy all dependencies in $\mathcal{R}$. The set of all instances of $\mathcal{R}$ is denoted $I(\mathcal{R})$.

A relational table is in *third normal form* (3NF) if whenever $X \rightarrow A$ holds in R and A is not in X, then either X is a key or A is a member of a key for R. If a relational table is in 3NF and has only simple keys, then in all functional dependencies $X \rightarrow Y$ where $Y \not\subseteq X$, the attributes X include a key attribute. Hence, the functional dependencies in these tables are fully characterized by the key dependencies. Date and Fagin showed that such relational schemas have a number of desirable properties [DF92].[6] They recommend the use of this class of relational schemas as a simple, easily understood way to ensure that a schema is free of commonly occurring anomalies.

6.2 Representing Relational Schemas in 3NF with Simple Keys

In this section, we consider the class of relational schemas where each relational table is in 3NF with only simple keys but which may be further constrained by bidirectional unary inclusion dependencies that force two attributes to contain the same set of values in any valid instance.[7] We call such schemas *simple relational schemas.*

Algorithm 6.1 translates a simple relational schema into a simple SIG schema with equivalent information capacity. Let $\mathcal{R}$ be a simple relational schema containing relation schemas $R_1, R_2, ...R_r$. We treat all attributes equated by inclusion dependencies as a single attribute (and perform the appropriate renaming). Let a_i be the number of attributes in R_i. The SIG translation of $\mathcal{R}$, denoted $S_{\mathcal{R}}$, is constructed as follows. A node is created for each attribute. Since all keys are unary, every relation must have at least one attribute that is a key. We select one key attribute and call it A_1. For each additional attribute A_i, we place an edge from A_1 to A_i. If A_i is also a key then the edge is bijective. Otherwise, the edge is a total, surjective, function. Formally, $S_{\mathcal{R}} = (G_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}})$ is defined by the following algorithm.

Algorithm 6.1 Input: a simple relational schema $\mathcal{R}$.

Output: an equivalent simple SIG schema $S_{\mathcal{R}}$.

```
For i = 1 to r
    Let $R_i = \{A^i_1, A^i_2, ..., A^i_{a_i}\}$ where $A^i_1$ is a key.
      For j = 1 to $a_i$
          Let $N^i_j \in N_{\mathcal{R}}$ be a node corresponding to $A^i_j$ and let $\tau(N^i_j) = \tau(A^i_j)$.
      For j = 2 to $a_i$
          Let $e^i_j : N^i_1 \rightarrow N^i_j \in E_{\mathcal{R}}$.
          If $A^i_j$ is a key then let $\mathcal{A}_{\mathcal{R}}(e^i_j) = \{f, i, s, t\}$. Otherwise, let $\mathcal{A}_{\mathcal{R}}(e^i_j) = \{f, s, t\}$.
```

Theorem 6.1 If $\mathcal{R}$ is a simple relational schema then the SIG translation $S_{\mathcal{R}}$ created by Algorithm 6.1 has equivalent information capacity ($\mathcal{R} \sim_{abs} S_{\mathcal{R}}$). •

[6]For example, such schemas are necessarily in projection-join (fifth) normal form.

[7]We only consider bidirectional inclusion dependencies to simplify the translation algorithm we present. The algorithm is easily extended to handle all unary inclusion dependencies.

Proof We show equivalence by constructing an equivalence preserving mapping $f : I(\mathcal{R}) \rightarrow I(S_{\mathcal{R}})$. Let $\Im \in I(\mathcal{R})$. The instance $\Im$ assigns a set of data values to each attribute of each relational table. The function f creates an instance $f(\Im) \in I(S_{\mathcal{R}})$ that assigns that same set of data values to each node corresponding to an attribute of $\mathcal{R}$. Then, for each tuple $(a_1, a_2, ..., a_{a_i})$ of a table R_i, the pair (a_1, a_j) is placed in $f(\Im)[e_j]$ for $2 < j \leq a_i$. Formally, f is defined as follows.

```
For i = 1 to r
    Let $R_i = \{A_1^i, A_2^i, ..., A_{a_i}^i\}$ where $A_1^i$ is a key.
     For j = 1 to $a_i$
         Let $f(\Im)[N_j^i] = \Im[A_j^i]$.
     For a = 2 to $a_i$
         Let $f(\Im)[e_a^i : N_1^i - N_a^i] = \Im[\{A_1^i, A_a^i\}]$.
```

The key dependencies on R_i guarantee that each edge will satisfy its annotations so $f(\Im)$ is a valid instance of $S_{\mathcal{R}}$. Hence, f is a well defined total function from $I(\mathcal{R})$ to $I(S_{\mathcal{R}})$.

We now show that f is injective. Suppose $\Im 1$ and $\Im 2$ are two distinct instances of $I(\mathcal{R})$. Without loss of generality, assume $\Im 1[R_i]$ contains a tuple $(a_1^i, a_2^i, ...a_{a_i}^i)$ that is not in $\Im 2[R_i]$. If $a_1^i \notin \Im 2[A_1^i]$ then clearly $f(\Im 1) \neq f(\Im 2)$. Suppose $a_1^i \in \Im 2[A_1^i]$, so there is a tuple $(a_1^i, b_2^i, ..., b_{a_i}^i) \in \Im 2[R_i]$. Since $(a_1^i, a_2^i, ...a_{a_i}^i)$ is not in $\Im 2$, there is some $j \neq 1$ such that $b_j^i \neq a_j^i$. By the definition of f, $(a_1^i, a_j^i) \in f(\Im 1)[e_j^i]$ and for no $a \neq a_j^i$ is $(a_1^i, a) \in f(\Im 1)[e_j^i]$. Similarly, $(a_1^i, b_j^i) \in f(\Im 2)[e_j^i]$ and for no $b \neq b_j^i$ is $(a_1^i, b) \in f(\Im 2)[e_j^i]$. Therefore $f(\Im 1) \neq f(\Im 2)$. Hence, f is injective.

To see that f is surjective, let $\Im 2$ be any instance of $S_{\mathcal{R}}$. We can construct and instance $\Im 1$ of $\mathcal{R}$ as follows. For each $a_1^i \in \Im 2[N_1^i]$, where $(a_1^i, a_j^i) \in \Im 2[e_j^i]$, let $(a_1^i, a_2^i, ..., a_{a_i}^i) \in \Im 1[R_i]$. Clearly, $\Im 1$ is a valid instance of $\mathcal{R}$, and $f(\Im 1) = \Im 2$. Hence, f is surjective.

The bijection $f : I(\mathcal{R}) \rightarrow I(S_{\mathcal{R}})$ is an equivalence preserving mapping, so $\mathcal{R} \sim_{abs} S_{\mathcal{R}}$. □

6.3 An Example Translation

We now turn to an example illustrating the translation discussed above. It is known that the relational model, without dependencies, permits no data relativism in that two relational schemas are equivalent if and only if they are identical (up to renaming of attributes and relations) [Hul86]. We use an example that shows this is not the case for the class of relational schemas considered in Section 6.2.

In Schema $R1$ of Figure 2, each of the three attributes is a key. Schema $R2$ is a decomposition of $R1$ representing similar information. Despite the fact that $R2$ is a dependency preserving decomposition of $R1$ with the lossless join property, it has strictly greater information capacity than $R1$. This can be seen by examining the equivalent SIG schemas $S1$ and $S2$ that are depicted in Figure 3. The closure of $S1$ is $S1$ and the closure of $S2$ is depicted in Figure 3. The set of valid instances of $S2$ and $S2^*$ are identical so $S2 \sim_{abs} S2^*$. The schema $S2$ absolutely dominates $S1$ (via an information capacity preserving mapping f that

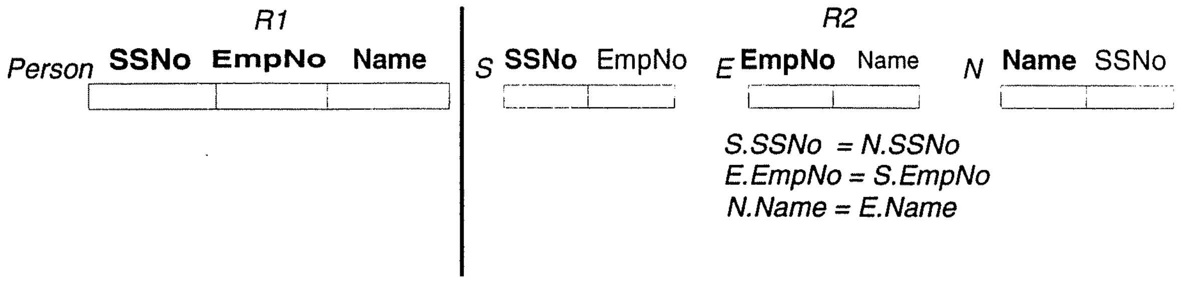

Figure 2: Two example relational schemas in 3NF. Keys are depicted in bold.

carries instances of the three nodes and edges e_1 and e_2 in $S1$ to instances of the corresponding constructs in $S2$ and populates e_3 with any bijection, for example $e_2 \circ e_1^\circ$). However, $S2$ has strictly more information capacity than $S1$. In $S1$, only a single binary relation between EmpNo and Name can be represented (via the composition of the edges e_1 and e_2). This reflects the fact that a single table in the relational model can only represent a single relationship between any subset of the attributes. However, in $S2$ (and $R2$), two different binary relations between EmpNos and Names are represented.

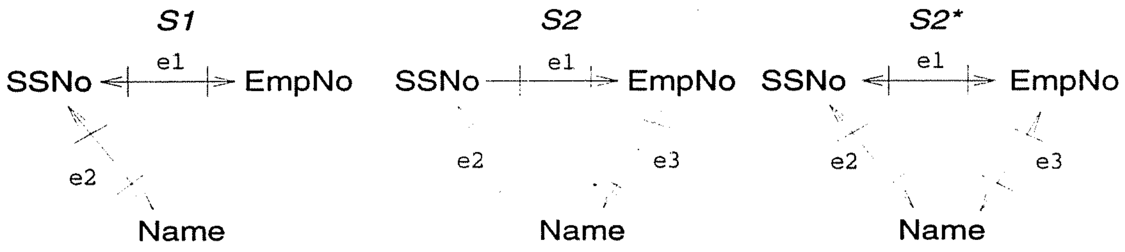

Figure 3: The SIG translations of R1 and R2.

Consider the addition to $R1$ of another relational table T with two attributes, EmpNo and Name, where EmpNo is a key and the inclusion dependencies $T[EmpNo] = Person[EmpNo]$ and $Person[Name] = T[Name]$ hold. The closure of the SIG translation of this augmented schema is isomorphic to $S2^*$ and therefore absolutely equivalent to $R2$. Hence, the augmented $R1$ and original $R2$ have equivalent information capacity. This fact is not at all obvious from the original relational schemas themselves. By developing algorithms to determine the absolute dominance or equivalence of SIG schemas, we can reason about the dominance or equivalence of schemas in the relational or other data models.

The specific example we chose for this section also illustrates an important point about schema decomposition. $R2$ is a dependency preserving decomposition of $R1$ with the lossless join property. Such decompositions are considered benign and even desirable in database design to avoid, among other things, redundancy in data and update anomalies.[8] However, the question of when a decomposed schema has equivalent information capacity to the original schema is typically overlooked.

[8]Granted that $R1$ is already in projection-join normal form so decomposition would likely not be applied in this example.

6.4 Limitation of the SIG model

As we have just demonstrated, the constraints included in the SIG model are sufficient to capture large subsets of the relational model. However, SIG schemas cannot model the complete class of relational schemas with functional dependencies. To illustrate this, consider the schema R shown in Figure 4.

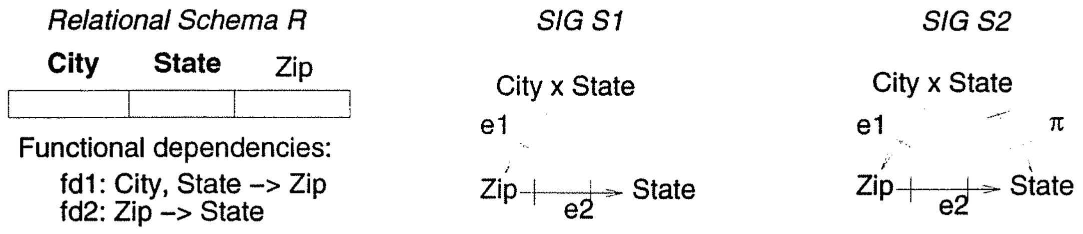

Figure 4: A relational schema that cannot be captured by an equivalent SIG.

As noted above, a single relation in the relational model (or a set of relations to which the universal relation assumption is applied) can only model a single relationship between a given set of attributes. This property has been called the *relationship uniqueness assumption* or RUA [AP82]. Hence, the two functional dependencies expressed on the relational schema of Figure 4 are not independent. A pair of city-state values (c1,s1) functionally determines a single zip code value z1. This zip code in turn functionally determines a single state value. This value, however, is constrained by the RUA to be the value s1.

The RUA is not made in the SIG model and, in fact, cannot be expressed with the limited constraints included in the model. Figure 4 depicts the simplest and most intuitive SIG schemas that dominate the relational schema. The first captures as simple edges the two functional dependencies of the relational schema. However, there is no constraint that for any city-state value pair, $(c1, s1)$, the equality $e_2 \circ e_1(c1, s1) = s1$ holds. There is an implicit total surjective function between city-state pairs and state values representing the projection of state values. However, even if this function is made explicit by including the projection edge π (as in Schema $S2$), there is still no constraint that the composition of e_1 followed by e_2 must equal π. Hence, these SIG schemas have strictly greater information capacity than the relational schema R.[9]

In fact, we can argue informally that any SIG schema that dominates R must include a functional path from a node containing city and state values to a node for zip codes. There must also be a functional path from the zip code node to the state node. Since there is no way to constrain the composition of the two functional paths to equal the projection of state values, no SIG schema that dominates R can be dominated by it.[10]

This example points out that cardinality and existence constraints alone do not capture the more elaborate constraints such as constraints on the equality of relationships. Such constraints are implicit within the relational and other models.

[9] A formal proof can be constructed by examining the relative sizes of $I(R)$ and $I(S1)$.
[10] At least through the intuitive mappings we are considering in this informal discussion.

7 Future Directions

This work is only one component of our efforts to develop formal but practical solutions to schema translation and integration problems. A central theme of our work is the development of efficient algorithms to test for the information capacity dominance and equivalence of SIG schemas [MIR94]. Additionally, we will develop equivalence preserving translations from existing data models into the SIG model and use our dominance and equivalence testing algorithms to understand the relative information capacity of schemas expressed in different data models [MIR93a, MIR93b]. Finally, the ultimate goal of our research is to develop a useful tool based on formal principles that aids in schema translation and integration.

References

[AH88] S. Abiteboul and R. Hull. Restructuring Hierarchical Database Objects. *Theoretical Computer Science*, 62:3–38, 1988.

[AP82] P. Atzeni and D. S. Parker. Assumptions in Relational Database Theory. In *Proc. of the ACM Sym. on Principles of Database Systems*, pages 1–9, Los Angeles, CA, March 1982.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice Hall, New York, NY, 1990.

[Dav73] M. Davis. Hilbert's Tenth Problem is Unsolvable. *American Mathematical Monthly*, 8(3):233–269, March 1973.

[DF92] C. J. Date and R. Fagin. Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases. *ACM Transactions on Database Systems*, 17(3):465–476, September 1992.

[Hul86] R. Hull. Relative Information Capacity of Simple Relational Database Schemata. *SIAM Journal of Computing*, 15(3):856–886, August 1986.

[Hul87] R. Hull. A Survey of Theoretical Research on Typed Complex Database Objects. In J. Paredaens, editor, *Databases*, chapter 5, pages 193–256. Academic Press, London, U.K., 1987.

[HY84] R. Hull and C. K. Yap. The Format Model: A Theory of Database Organization. *Journal of the ACM*, 31(3):518–537, 1984.

[MIR93a] 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*, pages 120–133, Dublin, Ireland, August 1993.

[MIR93b] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Understanding Schemas. In *Proceedings on Research Issues on Data Engineering: Interoperability in Multidatabase Systems*, pages 170–173, Vienna, Austria, April 1993.

[MIR94] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice. *Information Systems*, 19(1):3–31, 1994.

[MS92] V. M. Markowitz and A. Shoshani. Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach. *ACM Transactions on Database Systems*, 17(3):423–464, September 1992.

[OY82] C. Ó'Dúnlaing and C. K. Yap. Generic Transformation of Data Structures. In *Sym. on Foundations of Computer Science*, pages 186–195, Chicago, IL, November 1982.

[RR87] A. Rosenthal and D. Reiner. Theoretically Sound Transformations for Practical Database Design. In *Proc. of the Int'l Conf. on Entity-Relationship Approach*, pages 115–131, New York, NY, November 1987.

[RR93] A. Rosenthal and D. Reiner. Tools and Transformations - Rigorous and Otherwise - For Practical Database Design. Technical report, MITRE Corp., February 1993.

[Tar72] R. E. Tarjan. Depth First Search and Linear Graph Algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.