Restricting Is-A Related Groupings Using Object Equivalence

Elke A. Rundensteiner, Lubomir Bic, Jonathan Gilbert, and Meng-Lai Yin
Department of Information and Computer Science
University of California
Irvine, CA 92717

Abstract: Most research on semantic integrity has taken place in the traditional database fields; specifically, the relational data model. Advanced models, such as, semantic and object-oriented data models, have developed higher-level abstractions to increase their expressive power in order to meet the needs of newly emerging application domains. This allows them to incorporate some semantic constraints directly into their schemas. There are however many types of restrictions that cannot be expressed solely by these high-level constructs. Therefore we extend the potential of advanced models by augmenting their abstractions with useful set restrictions. In particular, we exploit the notion of *object identity* for the definition of a dual model of *object equivalence* for complex objects constructed by *grouping abstractions*. Based on this, we define a number of restrictions that control the structure and composition of a *semantic set grouping*. This framework of restrictions is then extended to *is-a related set groupings*. This permits each set grouping to capture more subtle distinctions of the concepts in the application environment.

1 INTRODUCTION

In the past decade research into the design and implementation of semantic database models has risen to prominence. Semantic models attempt to cope with the demands of new sophisticated database applications by modeling their environment more directly than traditional database models. This trend has been clearly reflected in the literature from the development of the Entity-Relationship model [3] and the Hierarchical Relational model [16], to the advent of the first *semantic database models* [4, 6, 15]. The evolution continues today (see, for example, [2, 1, 7, 11, 9, 13]). Important features common to most of these models are a property inheritance hierarchy (is-a relationship), a decomposition hierarchy (is-part-of relationship), and a variety of powerful constructs for collecting related data, referred to as *semantic groupings*. The latter allow new sets of database entities to be constructed from the stored data. The most common groupings are based on extended definitions of sets, power sets, and the Cartesian product.

The work presented in this paper consists of several parts. First, we formalize the definition of several basic types of semantic groupings. An extended notion of a set (called *set grouping*) is defined by allowing elements to repeat and an order to be imposed on its elements. The *is-a related set groupings*, *the power set grouping* and the *Cartesian grouping* are then defined based on the set grouping definition.

Then, we incorporate the concept of *object identity* into our framework. The notion of object identity plays an important role in many of today's object-oriented models, but has been exploited to a lesser degree by semantic data models, which may only include it in an implicit form. In this paper, we exploit the notion of object identity for the development of semantic constraints. We develop a dual model of *object equivalence* for complex objects constructed by semantic groupings, based on *identity* and *value*. Objects may be: (1) distinct and distinguishable, (2) distinct but indistinguishable (shallow equal), (3) the same but distinguishable (when viewed as participating in different classes), or (4) the same and indistinguishable (deep equal) [8, 12, 14]. Our paper extends the conventional set-oriented notation with the notion of object identity and the ability to handle complex objects and classes. The combination of these different forms of object equivalences with the different types of semantic groupings allows us to formulate a framework of structural constraints which is currently unique.

Lastly, we present a number of restrictions that can be imposed on the basic set grouping and on the is-a related set groupings. The extension of the proposed framework of restrictions to the Cartesian and the power set groupings is described in another paper [12]. A semantic grouping can be specialized to suit the particular concept to be modeled by selecting the appropriate restrictions. We only consider set-related restrictions, i.e., restrictions imposed on the structure of the groupings or their population. Other types of general constraints, such as attribute- or value-based ones, are not considered here. The purpose of the proposed framework is twofold: First, it increases the modeling power of the

DATABASE SYSTEMS FOR ADVANCED APPLICATIONS '91
Ed. A. Makinouchi

paradigm and second, it serves as a basis for integrity constraint management [5, 10].

The paper is organized as follows. In Section 2 we define the basic types of semantic groupings. *Object equivalence* in the context of these groupings is discussed in Section 3. Section 4 then presents the restrictions on basic set groupings, while Section 5 extends these restrictions to is-a related set groupings. The concepts presented in Sections 4 and 5 are illustrated by a set of examples. Conclusions are presented in the final section.

2 SEMANTIC GROUPINGS

2.1 Classes, Types, and Sets

In semantic data modeling, the notions of set, class, and type are not always clearly distinguished. In this paper we take the following position. Entities in our data model represent a concept (concrete or abstract) in the application world. The term entity is used in this paper in its most generic form — it may, for example, stand for a row in a relation table [4], an entity (or a relationship) in the entity-relationship model [3], an entity in a semantic data model [7] or an object in an object-oriented model [11, 9]. Entities in the world of the application are grouped into *classes*, based on common properties. The class notion, however, serves a dual purpose. It is a generic description of all entities which belong to that class and hence it imposes a *type* on those entities. At the same time a class represents the *set* of entities which conform to its generic description (type). This distinction is important because certain attributes only apply to a set as a whole (e.g., cardinality) while others are applicable to its individual elements (e.g., color). In this paper we are interested primarily in the set-aspect of a class and, therefore, will be referring to sets rather than classes.

2.2 Set Groupings

A mathematical set is a collection of elements (entities) taken from a given domain of discourse. The domain for each set depends on the particular application being modeled. We will extend the notion of a set as follows:

1. elements may have their own *identity*,
2. multiple *non-distinguishable* copies of an element may occur, and
3. an order may be imposed on the elements.

Each of these extensions will be described in the following subsections. Note that groupings which adhere to these three extensions are called *set groupings*.

The term *identity* is introduced to define a time-invariant identifier of an entity, that is independent from the state (value) of the entity. It is possible for two entities to have exactly the same state and thus be *indistinguishable* from one another, without actually being the *same* entity. Our model allows entities ("copies") that are *indistinguishable* based on their state but *distinct* based on their identifier to be in the same set grouping.

We distinguish between *simple* and *constructed* entities. Simple entities are taken directly from a base domain of the application while constructed entities are built from other entities using Cartesian or power set aggregation abstractions; both of these will be discussed at the end of this section. An example of a simple entity may be an integer or a string as well as complex objects in the real world, like for instance, a Person or a Ship. Constructed entities are composed from other entities of the database. This then leads to two classes of set groupings, namely, those that contain simple elements and those that contain constructed elements. We define a *base-set grouping* to be a collection of simple entities. All other *set groupings* are based on already existing groupings, each of which may be constructed in one of the three ways as described below.

2.3 Is-A Related Set Groupings

The first type of abstractions is concerned with *is-a* related set groupings, in particular, *subset* and *union* groupings. These *is-a* related set groupings are closely related to the generalization/specialization abstractions. Two sets A and B are said to be *is-a* related if every element from A is described by all the properties which elements in B have. For example, the sets Red Cars and Cars are *is-a* related, since every red car has all the properties of a car (plus some additional ones). This is generally referred to as property inheritance. These new set groupings are constructed by collecting some elements of existing set groupings without changing the shape or identity of these elements. A *subset grouping* is based on the mathematical definition of a subset c which consists of some elements of an existing set. We employ a more general subset definition where a subset grouping S is based on one *or more* existing set groupings $S_1, S_2, ..., S_m$. Each element s of the subset grouping S must exist in all these underlying set groupings S_i for i =

1 to m. More precisely, $S \subseteq (S_1, S_2, ..., S_m)$ if and only if $((\forall i, 1 \leq i \leq m)(s \in S \Rightarrow s \in S_i))$. Note that the above definition of a subset is different from a set intersection. The latter corresponds to the *largest* possible subset, i.e., a special case of a subset. This definition is necessary to maintain the *is-a* relationship between the subset S and all set groupings it is based on. The following implication holds, $S \subseteq (S_1, S_2, ..., S_m)$ implies $((\forall i, 1 \leq i \leq m)(S \textit{ is-a } S_i))$.

The *union grouping* is an abstraction which forms a possibly heterogeneous set grouping U from several existing set groupings $S_1, S_2, ..., S_m$. The union grouping is denoted by $U = \uplus_{i=1}^{m} S_i$. This construct collects the elements from all involved set groupings into one new grouping. The result of a *union grouping* is again a set grouping. More precisely, $U = \uplus_{i=1}^{m} S_i$ if and only if $((\forall\ i,\ 1 \leq i \leq m)\ (s \in S_i \Rightarrow s \in U))$ and $(s \in U \Rightarrow ((\exists\ i,\ 1 \leq i \leq m)\ s \in S_i))$. In other words, if and only if an entity s belongs to any of the underlying set groupings S_i, then s must also belong to the union U. Again, the *is-a* relationship is guaranteed, namely, $U = \uplus_{i=1}^{m} S_i$ implies $((\forall\ i,\ 1 \leq i \leq m)\ (S_i \textit{ is-a } S))$.

The domain of a subset grouping is the intersection of the domains of all underlying set groupings. The domain of the union grouping is the union of the domains of all underlying set groupings.

2.4 The Cartesian Aggregation Grouping

A *Cartesian aggregation grouping* ([16]) is an abstraction which allows a relationship between several database entities to be viewed as a single aggregate or *complex* entity. For example, a relationship between a person, a hotel room and a date can be viewed as a reservation. Each element in the Cartesian grouping is taken from the *cross product* of existing set groupings and a new unique identity is associated with it. A Cartesian grouping C based on the set groupings S_1, S_2, ..., S_n is defined by $C \subseteq S_1 \times S_2 \times ... \times S_n$. We say that set S_1 fills *position* 1 in the grouping, set S_2 fills *position* 2, etc. The ordering of positions is not essential to the model, it is just a matter of notational convenience. In fact, the positions p_i are usually referred to by labels unique to C which represent the role that the set grouping plays in the Cartesian grouping. If L_i is the label chosen for the position p_i, respectively, then we denote $C \subseteq (L_1 : S_1) \times (L_2 : S_2) \times ... \times (L_n : S_n)$. The previous example would be denoted by reservation $\subseteq$ (who: person) $\times$ (where: hotel-room) $\times$ (when: date).

The notion of a domain for a Cartesian grouping is defined as follows. For all i, let D_i be the domain of S_i. Then, the domain D of C corresponds to the cross products of the domains of the set groupings underlying C, i.e., $D = D_1 \times D_2 \times ... \times D_n$.

An element t of a Cartesian grouping C is an aggregate entity consisting of n components where the i^{th} component is taken from the set grouping underlying the i^{th} position, S_i. We refer to the i^{th} component of t by $t[L_i]$ where L_i is the label of the i^{th} position, or simply, t[i]. Of course, $t[L_i] \in S_i$. To refer to several positions of an aggregate entity at the same time, we use the notation $t[L_{i1}, L_{i2}, ..., L_{ik}]$ where $L_{i1}, L_{i2}, ..., L_{ik}$ are distinct labels associated with *some* of the set groupings S_i. This is called an aggregate projection. Similarly, $C[L_{i1}, L_{i2}, ..., L_{ik}]$ refers to the collection of all aggregate projections of the Cartesian grouping C using labels $L_{i1}, L_{i2}, ..., L_{ik}$. More formally, $C[L_{i1}, L_{i2}, ..., L_{ik}] := \{ t[L_{i1}, L_{i2}, ..., L_{ik}] \mid t \in C \}$ where $t[L_{i1}, L_{i2}, ..., L_{ik}]$ retains the identity of the aggregate entity t. The latter generalizes the notion of a projection in the relational model.

2.5 The Power Set Grouping

A *power set grouping* is based on the mathematical concept of a power set of a set S which consists of all subsets of S. A power set grouping, denoted by S^*, is an abstraction based on a set grouping S. Each element of S^*, which corresponds to a subset grouping of s, has its own unique identity. Hence S^* consists of some (or all) of the possible subsets of S. In general, an element of S^* models a single (complex) entity, usually referred to as a *cover aggregate*. It should be emphasized that the power set abstraction forms a set of aggregates each of which contains a set of elements from S. For example, clubs are cover aggregates, each composed of a set of people. We distinguish between power set groupings whose elements have a significant order and those power set groupings which do not have a significant order. In power set groupings S^* where the elements have a significant order, each permutation of a subset $s \in S^*$ is considered to be distinct with its unique identity. In a power set grouping without a significant order defined for its elements, all permutations of a subset $s \in S^*$ are considered to be equivalent. In our model, a power set grouping can be defined, based on any existing set grouping which is not already a power set grouping.

The domain of a power set grouping S^* derived from the set grouping S is defined as follows. If D is the domain of S, then the domain of S^* is defined to be D as well.

3 OBJECT EQUIVALENCE

3.1 A Dual Model of Object Equivalence

The concept of identity has been introduced implicitly in some semantic models (explicitly in object-oriented systems [8]) to better cope with complex entities. We use it explicitly: an entity (with identity) consists of two parts — an *identity* and a *state*. Let us denote the identity and the state of an entity s by s.id and s.state, respectively. The identity is time-invariant, i.e., when the state of an entity is modified its identity is unchanged. This is typically implemented by a surrogate — a system-assigned global ID invisible to the user. The identity of a given entity is independent of its current state. Hence it is possible for two entities to have exactly the same state and thus be indistinguishable from one another, without actually being the *same* entity. Note that this is in sharp contrast with mathematical sets, which do not have the concept identity associated with their elements. In a mathematical set each element is essentially the string of symbols used to represent it. When the element is modified (i.e., a symbol is changed), it becomes a *different* element. These observations lead us to the following definitions:

Definition 1: Let s_1 and s_2 be two elements *with identity* from the set groupings S_1 and S_2 respectively. The relationship $s_1 \doteq s_2$ (pronounced *dot-equal* or *identical*) holds if and only if s_1 and s_2 represent the *same entity*, i.e. s_1.id = s_2.id.

Note that some simple entities may not have identities. In this case, the predicate $\doteq$ is based on the state of the entities instead of their identities. Then, the relationship $s_1 \doteq s_2$ holds if and only if s_1 and s_2 represent the *same value*, i.e. s_1.state = s_2.state.

To capture the concept of indistinguishability, we need to consider Cartesian aggregates and cover aggregates separately.

The next two definitions apply to Cartesian aggregates:

Definition 2: Let c_1 and c_2 be two Cartesian aggregates in $C \subseteq S_1 \times S_2 \times ... \times S_n$ with $c_1 = [s_1, s_2, ..., s_n]$ and $c_2 = [s'_1, s'_2, ..., s'_n]$. Then, c_1 and c_2 are called *component-identical*, denoted by $c_1 =^c c_2$, if and only if their components are pairwise identical, i.e., the following holds: $(\forall i)$ $(s_i \doteq s'_i)$.

Definition 3:

1. Let c_1 and c_2 be Cartesian aggregates as in Definition 2. Then, c_1 and c_2 are *value-indistinguishable*, denoted by $c_1 =^v c_2$, if and only if their components are pairwise *value-indistinguishable*, i.e., the following holds: $(\forall i)$ $s_i =^v s'_i$.
2. Let c_1 and c_2 be simple entities. Then, c_1 and c_2 are *value-indistinguishable*, denoted as $c_1 =^v c_2$, if and only if they have the same state, i.e., c_1.state = c_2.state.

Note that the last definition is recursive; it stops when applied to simple entities.

To illustrate these two definitions, which are based on the work of Khoshafian and Copeland [8], consider a situation where a new car, say car_2, has been built out of the major parts of another (perhaps damaged) car, say car_1. Car_1 and car_2 have different identities but if only the major parts are recorded by the model, they are indistinguishable. According to Definition 2, we refer to entities that share the same components as *component-identical*. On the other hand, two cars could be indistinguishable by virtue of having the same type of body, engine, and wheels and being painted with the same color. Their components, however, would be physically distinct entities (i.e., have different identities). According to Definition 3, we refer to such entities as *value-indistinguishable* (or just *indistinguishable*, for short). We now present analogous definitions of the relations $=^c$ and $=^v$ for cover aggregation elements (of power set groupings).

Definition 4: Let c_1 and c_2 be two cover aggregation elements with $c_1 = \{s_1, s_2, ..., s_n\}$ and $c_2 = \{s'_1, s'_2, ..., s'_n\}$. Then, c_1 and c_2 are called *component-identical*, denoted by $c_1 =^c c_2$, if and only if they have the same cardinality (denoted by $|c_i|$) and their elements are pairwise identical. In other words, the predicate $=^c$ evaluates to true if and only if the following holds:

1. $|c_1| = |c_2|$, and
2. if both c_1 and c_2 have a significant order then $s_i \doteq s'_i$ for all i. If c_1 and c_2 do not have a significant order then there is an ordering of elements of c_1 and c_2, for instance, $c_1 = \{s_{i_1}, s_{i_2}, ..., s_{i_n}\}$ and $c_2 = \{s_{i_{1'}}, s_{i_{2'}}, ..., s_{i_{n'}}\}$, such that $s_{i_j} \doteq s_{i_{j'}}$ for all j. If, without loss of generality, c_1 does have a significant order, and c_2 does not, then there is an ordering of the elements of c_2, for instance, $c_2 = \{s_{i_{1'}}, s_{i_{2'}}, ..., s_{i_{n'}}\}$, such that $s_i \doteq s_{i_{j'}}$ for all i.

Definition 5: Let c_1 and c_2 be as in the previous definition. Then, c_1 and c_2 are called *value-indistinguishable*, denoted by $c_1 =^v c_2$, if and only if they have the same number of elements and their components are pairwise *value-indistinguishable*. The predicate $=^v$ evaluates to true if and only if the following holds:

1. $|c_1| = |c_2|$, and
2. if both c_1 and c_2 have a significant order then $s_i =^v s'_i$ for all i. If c_1 and c_2 do not have a significant order then there is an ordering of elements of c_1 and c_2, for instance, $c_1 = \{s_{i_1}, s_{i_2}, ..., s_{i_n}\}$ and $c_2 = \{s_{i_{1'}}, s_{i_{2'}}, ..., s_{i_{n'}}\}$, such that $s_{i_j} =^v s_{i_{j'}}$ for all j. If, without loss of generality, c_1 does have a significant order, and c_2 does not have a significant order, then there is an ordering of the elements of c_2, for instance, $c_2 = \{s_{i_{1'}}, s_{i_{2'}}, ..., s_{i_{n'}}\}$, such that $s_i =^v s_{i_{j'}}$ for all i.

The last definition is again recursive; it stops when applied to simple elements.

To illustrate the above two definitions, imagine two cover aggregate entities $race_1$ and $race_2$ which model the sets of cars participating in a certain car race. Assume that $race_1$ and $race_2$ refer to two different car races, but that exactly the same cars participate in both. By just looking at the participating cars one would not be able to distinguish between these two sets of cars and hence they would be *component-identical* (by Definition 4).

On the other hand, there could be two races in which different cars take part, but these cars pairwise look alike. If for each $race_r$ $(1 \leq r \leq 2)$ there exist an ordering $(car_{r,1}, ... car_{r,n})$ such that $car_{1,i}$ is indistinguishable from $car_{2,i}$ for $1 \leq i \leq n$, then these two cover aggregates (races) are *value-indistinguishable* by Definition 5.

Definition 6: For simple entities s_1 and s_2, the two predicates $s_1 =^v s_2$ and $s_1 =^c s_2$ are the same and default to the notion of taking on the same value.

Hence, simple entities are either both component-identical and value-indistinguishable or neither. This is because simple entities have no components. Both predicates evaluate to true if and only if s_1.state = s_2.state.

The above definitions capture the important property of the real-world, namely, that many *different* entities may have the *same* attributes and external appearance. Furthermore, even if two initially distinguishable entities evolve over time so that they become indistinguishable, their identities as two separate individuals will be preserved.

3.2 Multiple Element Occurrences

In the following discussion, we use the term indistinguishability for both *component-identity* and *value-indistinguishability* and we denote it by the symbol =. Although multiple indistinguishable elements may occur within a given set grouping each identity is unique and persistent. In other words, given two simple elements s_1 and s_2, the relation $s_1 = s_2$ may be true or false, depending on whether s_1 and s_2 are distinguishable or not. However, for distinct entities with identity the relation $s_1 \doteq s_2$ is always false within a single set grouping. Every element in a set grouping has an identity which is distinct from all other elements in that grouping, therefore, for any two elements with identity the predicate $\doteq$ is always false within *one* set grouping[2]. In the remainder of this section, we study interrelationships between the different types of predicates within one set grouping.

Lemma 1: Within a given set grouping S the following holds:

(1) $s1 \doteq s2 \Rightarrow s1 =^c s2$, and

(2) $s1 =^c s2 \Rightarrow s1 =^v s2$.

And by (1) and (2),

(3) $s1 \doteq s2 \Rightarrow s1 =^v s2$.

Lemma 1 establishes that identical entities always consist of exactly the same components and thus look alike within one set grouping (a natural phenomenon of the world). Also, entities which share the same components are indistinguishable.

Next, we present conclusions that can be drawn when allowing/disallowing indistinguishable elements in a set grouping.

Lemma 2: If no component-identical elements are allowed in a set grouping S, then not only (1), (2) and (3) from Lemma 1 hold in S but also:

(4) $s1 \neq s2 \Rightarrow s1 \neq^c s2$.

By (1) and (4), we get:

(5) $(s1 \neq s2) \iff (s1 \neq^c s2)$, or, equivalently, $(s1 \doteq s2) \iff (s1 =^c s2)$.

Nothing can be concluded, however, from the fact $s1 \neq^v s2$.

[2]This is based on pragmatic grounds. There is nothing fundamental that would prevent us from allowing multiple occurrences of the same element within a set, however, we did not find any concrete application where this would be necessary.

Lemma 3: If no (value-)indistinguishable elements are allowed in a set grouping S, then not only (1), (2) and (3) from Lemma 1 hold in S but also:

(6) $s1 \neq s2 \Rightarrow s1 \neq^v s2$, and, $s1 \neq^c s2 \Rightarrow s1 \neq^v s2$.

Hence, we have:

(7) $(s1 \doteq s2) \iff (s1 =^c s2) \iff (s1 =^v s2)$.

All eight combinations of truth values for the predicates $\doteq$, $=^c$ and $=^v$ are possible within the data model. This is because an entity (base entity as well as the component of an entity) may look differently when it is viewed as a member of different set groupings. For instance, a person has different characteristics when viewed as a student than as an employee; as a student sh/e may have a grade attribute while as an employee a salary attribute may apply. Therefore, it may be possible for ($element_1$ in employee-class) $\doteq$ ($element_2$ in student-class) to evaluate to true but ($element_1$ in employee-class) $=^v$ ($element_2$ in student-class) and/or ($element_1$ in employee-class) $=^c$ ($element_2$ in student-class) to evaluate to false. Lemma 1 indicates that within one set grouping such a situation cannot occur.

3.3 Ordering

The third extension to the standard definition of a set is to allow one or more orders to be imposed on the elements of a set grouping. For each order, this implies the existence of a predicate "$\leq$", which, when applied to any two elements of a set grouping ($s_1 \leq s_2$) returns true if s_1 precedes s_2 in the ordering and false otherwise. The specification of such an ordering takes one of two forms — an explicit enumeration or an evaluable function. An enumerated order requires that the relationship between elements be specified explicitly. For example, one could order the collection of cars by their price by explicitly listing them as "$car_1 \leq car_2 \leq ... \leq car_n$". At execution time, the boolean value for the predicate "$car_i \leq car_j$" (which can be interpreted as "car_i is less expensive than car_j") is obtained by examining the explicit enumeration. In the second case, a function is specified which derives the boolean value for "$\leq$" through some computation on entities or their components at execution time. Examples of such functions are the lexicographical order on words or the less-or-equal function defined on numeric values. For example, if cars were constructed entities with a price component, then one could order a collection of cars by their price. In that case, the less-or-equal function defined on numeric values would be applied to the price component of the respective cars at execution time and the relation *"$car_1 \leq car_2$"* would be evaluated by "car_1.price is less than car_2.price". Then, an explicit enumeration of all cars is no longer needed to specify such an order.

At most one of the orders defined on a set grouping can be designated as being *significant* or primary. If a set grouping has a significant order, then it will always be represented in that order. Furthermore, this order will be used when comparing the set grouping with other set groupings as described in Definition 4 and 5. To clarify the distinction between a significant and non-significant order, assume two set groupings A and B which contain the three alphabetic letters 'n', 't', 'o'. Note that A and B have the alphabetic order defined on them by which 'n' comes before 't', etc. This order is, however, not significant since two sets A = {'n', 't', 'o'} and B = {'o', 't', 'n'} are the same even if we do not list their elements in the same order. If, on the other hand, A and B are to represent the words 'not' and 'ton', then there is a significant order, namely, the order of enumeration, defined on them. In this case, A and B are no longer the same, since they represent two different words. Orders - whether significant or not - will be used in the remainder of this paper in the formulation process of constraints.

4 RESTRICTIONS ON SET GROUPINGS

4.1 Restrictions

We now present a set of restrictions that may be imposed on any set grouping, including Cartesian aggregation and power set groupings. By combining restrictions, distinct special cases of set groupings are produced, which capture a concept in the application more precisely. There are three kinds of user-specified restrictions based on: (1) cardinality, (2) the number of indistinguishable elements in a set, and (3) the range of element values with respect to a user defined order. All three restrictions are defined below. To simplify future discussions, we will use the term indistinguishable to mean *both* component-identical or value-indistinguishable except when explicitly stated otherwise.

1. The *cardinality* of a set grouping S (denoted by |S|) is the number of elements in S. The cardinality of a set grouping is determined by counting its distinct (not necessarily distinguishable) elements. By Lemma 1 an element may not occur more than once in a set grouping. Therefore the cardinality of a set grouping is easily determined. Cardinality can be restricted by giving an

integer range $[c_1{:}c_2]$ where $0 \leq c_1 \leq c_2$ and c_1 and c_2 are the lower and upper limit, respectively. If $c_1 = c_2$ then the set grouping S must have exactly c_1 elements at all times during its existence in the database. If $c_1 < c_2$ then the cardinality can vary within that range. The values $c_1 = 0$ and $c_2 = \infty$ impose no restrictions on the cardinality of the grouping and are assumed to be the default.

2. The *repetition* characteristic of a set grouping S specifies how many indistinguishable copies of an element may exist within S. The restriction has to be parameterized by $=^c$ and $=^v$ to indicate which type of indistinguishability is to be restricted. The repetition count can be limited by specifying a range $[r_1{:}r_2]$ where r_1 and r_2 are integers and $0 \leq r_1 \leq r_2$. This can be interpreted as: for any element s from the domain of S there exist at least r_1 and at most r_2 distinct elements of S which are indistinguishable from s (including s). If $r_1 = r_2$ with parameter $=^c$ or $=^v$ then for every element s in the set grouping there must be exactly r_1 elements s_i (i = 1, 2, ..., r_1) with $s =^c s_i$ or $s =^v s_i$. Recall that $s_i \doteq s_j$ is always false within a set grouping for $i \neq j$. The special case where $r_1 = r_2 = 1$ and the parameter is $=^v$ implies that no two elements in the set grouping are allowed to look alike. By Lemma 3, this also implies the implicit restriction of a repetition count of $[r_1, r_2] = [0,1]$ with parameter $=^c$. More general, by Lemma 3 any repetition restriction $[r_1, r_2]$ with the parameter $=^v$ enforces the upper bound of the range for parameter $=^c$ never to exceed r_2. Again, the unrestricted case, $r_1 = 0$ and $r_2 = \infty$ with either parameter, is assumed to be the default.

3. For a set grouping S with an order defined over its elements, a sequence of *ranges* $[l_1{:}u_1]$, $[l_2{:}u_2]$, ..., $[l_n{:}u_n]$ may be specified with respect to that order. Only elements which are contained within one of the ranges may appear in S. More precisely, an element s may appear in S if for one of the ranges $[l_i{:}u_i]$ $(1 \leq i \leq n)$ the following holds: $l_i \leq s \leq u_i$. Similarly, an open range $[l_i{:}u_i)$ can be specified which means that an element s may appear in S if the following holds: $l_i \leq s \leq u_i$. If no range is specified then the set grouping is unrestricted.

Since not all restrictions need to be specified for all sets, we use the following labels to designate which restriction is being referred to: (i) set cardinality, (ii) set repetition with $=^c$ or $=^v$, and (iii) ordering.

4.2 Examples of Set Groupings and their Restrictions

In this subsection examples of restricted set groupings are presented which demonstrate the usefulness of various combinations of these three basic restrictions.

Example 1:

A soccer team S can be characterized by:

1. set cardinality: [11:11].

   Interpretation: The cardinality of S is fixed to 11. The fact that any person can occur in S at most once, i.e., the same person cannot act as two or more team members, is automatically enforced by Definition 1. The repetition restriction does not apply, since we do not care whether two people look alike (value-indistinguishability) or possibly even share similar properities, such as, live at the same address (component-identity).

Example 2:

Let S model the collection of words in a dictionary. Its domain is the collection of character strings formed from a given alphabet. This set grouping can be characterized by the following:

1. set repetition with $=^v$: [0:1],
2. ordering: $[a : z^+]$.

   Interpretation: We assume that a a given string may occur at most once as a word in the dictionary, and since strings are simple entities without identity we may enforce this by the repetition restriction with the parameter $=^v$. Since all words are ordered alphabetically, the "smallest" word is the letter 'a' and the "largest" word is the infinite sequence 'z...z', denoted as z^+. This implies that no special symbols (like quotes or hyphens) would be allowed in our simple dictionary.

Example 3:

The conventional set S may be modeled as a special case of a set grouping by the following restrictions:

1. set repetition with parameter $=^v$: $[0 : 1]$

Interpretation: The cardinality of S can take any value from zero (empty set) to infinity, and thus no restriction is specified for it. The elements, taken from an underlying domain, are simple entities without identity; each may occur at most once in the set.

Note that example 1 could not be formulated unambiguously without the concept of object identity. This concept allows us to have two players in a team with exactly the same attributes. Example 2 controls repetition based on value equality as is done in value-based models, such as the relational model. In Example 3 we show that our framework subsumes the description of a conventional set. It shows that our framework subsumes the description of a conventional set.

While the restrictions presented in this section could be applied to any set grouping, there are certain additional restrictions that may be applied only in the case of Cartesian or power set groupings. These additional restrictions for Cartesian and the power set groupings are described in detail in another paper [12]. Furthermore, even the three basic restrictions may not always be applied freely in the case of is-a related set groupings. This is the topic of the next section.

5 RESTRICTIONS ON IS-A RELATED GROUPINGS

5.1 Restrictions

In this section, we discuss restrictions that may be imposed on non-base groupings that are part of the is-a hierarchy. We shall refer to these as *IS-A related set groupings.* Base set groupings do not depend on any other set groupings and thus the three restrictions discussed in the previous section can be imposed on them freely. The most common representatives of IS-A related groupings are *subset* and *union* groupings, which were introduced in Section 2. Such groupings are based on existing set groupings (through various derivation rules) and, consequently, additional constraints have to be met when applying the three set grouping restrictions to them. This is because there are strong interrelationships between IS-A set groupings. Below, we discuss the constraints on applying the three kinds of set grouping restrictions to subset and union groupings. For subsets, the following must hold:

1. The *cardinality* of a subset grouping S based on the sets S_i $(1 \leq i \leq m)$ must meet the following additional constraint. If $[c_{il}{:}c_{iu}]$ is the cardinality constraint for the set grouping S_i (for all i), then the cardinality constraint $[c_l{:}c_u]$ for the subset grouping S must satisfy the restriction $c_u \leq \min_{i=1}^{m} c_{iu}$. Hence, the cardinality of S is always less than or equal to the cardinality of the smallest set S_i. The lower bound is not restrained, and can take on any value between 0 and the upper bound.

2. The *repetition* characteristic of a subset grouping S, which specifies how many indistinguishable copies of an element may exist within S, must meet the following constraint. Let $[r_{il}{:}r_{iu}]$ be the repetition constraint with parameter $=^c$ $(=^v)$ for the set grouping S_i (for all i). Then the repetition count $[r_l{:}r_u]$ with parameter $=^c$ $(=^v)$ for the subset grouping S must satisfy the following: $(\forall$ i) ($r_u \leq r_{iu}$). The lower bound can again take any value between 0 and the upper bound.

3. If all underlying set groupings S_i have the same kind of order defined on them, then the subset grouping S *can* be ordered by the same ordering. In this case, the range restriction of the subset grouping consists of a subset of the intersection of the range restrictions of the underlying S_i. It is also possible to define additional orders and to further restrict S by specifying a range on the explicitly defined order.

The first restriction, called *cardinality* restriction, is constrained concerning the upper bound. This guarantees that the cardinality of S is always less than or equal to the cardinality of the smallest set S_i. Note that this is a powerful mechanism which may cause the non-base grouping (S) to force a restriction on the is-a hierarchy, i.e., the set groupings underlying S. It permits us to control the minimum number of elements the underlying set groupings must have in common. For example, if $|S_1|$ = n and $|S_2|$ = m and the lower cardinality bound of $S \subseteq (S_1, S_2)$ is k (with k $\leq$ n,m), then S_1 and S_2 must share at least k elements.

Let us now consider the characteristics and restrictions on union set groupings. By Lemma 1, an entity is only allowed to appear once in any set grouping. Therefore, an entity will occur only once in the union grouping S even if it occurs in more than one of the underlying set groupings. A union grouping may contain indistinguishable elements even if none of the set groupings participating in the union contain duplicates.

1. The *cardinality* constraint of the resulting union grouping S is determined from the cardinality constraints of the involved set groupings $S_1, S_2, ..., S_m$. Let $[s_{il} : s_{iu}]$ be the cardinality constraint for the set grouping S_i for all i = 1, ..., m. Then the cardinality constraint for S will be [l:u] with $l \geq \max_{i=1}^{m} s_{il}$, and $u \leq \sum_{i=1}^{m} s_{iu}$. This guarantees that the lower bound of the cardinality of S is not smaller than the cardinality of the set grouping with the largest lower bound and the upper bound is not greater than the sum of the maximal cardinalities of all set groupings.

2. The number of indistinguishable elements that appear in the union depends on the number of indistinguishable elements in the set groupings underlying the union. Let $[r_{il} : r_{iu}]$ be the repetition constraint on the set grouping S_i for all i = 1, ..., m. Then the constraint on S, denoted by [l:u], must be as follows: $l \geq \min_{i=1}^{m} r_{il}$, and $u \leq \sum_{i=1}^{m} r_{iu}$.

3. A union grouping can be ordered if all underlying set groupings are ordered by the same type of ordering. This means that all elements of the union have at least one attribute in common and the ordering is based on a common attribute. The range associated with the order is the union of the ranges of all set groupings.

Again note that the *cardinality* constraint is a very powerful constraint which permits us to control how many elements the underlying set groupings have in common. For example, if $|S_1|$ = n and $|S_2|$ = m and the upper cardinality bound of $S_1 \cup S_2$ is k, then S_1 and S_2 must share n + m - k elements. Hence, this constraint should be used carefully because it may result in a union which is so restricted that it will always be empty.

5.2 Examples of IS-A Related Set Groupings and their Restrictions

In this subsection examples of IS-A related set groupings are presented.

Example 1:

Let S model the collection of words in a dictionary as described in example 2 of Section 3. Then the set grouping Sub = {s | s is a word in a dictionary starting with the letter 'a'} is a subset grouping of S. It can be characterized by the following:

1. set repetition with $=^v$: [0:1],
2. ordering: $[a : b)$.

Interpretation: The fact that a given string may occur at most once as word in the dictionary is a restriction directly inherited from S. Since S is ordered, a range restriction can also be specified on Sub. In this case, all words of Sub must start with the letter 'a'.

Example 2:

Let S_{CS} and S_{ENG} be the set of all students enrolled in the Computer Science and Engineering departments, respectively. The corresponding cardinalities are $|S_{CS}| = 100$ and $|S_{ENG}| = 50$. Assume that students are ordered by their GPA. Let the set grouping S contain all students with a double major, i.e., those enrolled in the Computer Science as well as Engineering department and who have a grade point average of 3.5 and better. Then the set grouping S is a subset grouping of both groupings, i.e., S $\subseteq$ (S_{CS} , S_{ENG}). S could be constrained by:

1. set cardinality: [0 : 30]
2. ordering on GPA: [3.5 : 4.0]

Interpretation: The set grouping S contains only students who are in both underlying set groupings, S_{CS} and S_{ENG}. The upper bound on the cardinality of S is set to 30, which means that at most 30 students could have a double major in Engineering and Computer Science. Note that this imposes indirect constraints on inserting elements into the underlying sets. In particular, it would prevent the insertion of a new student into S_{ENG} or S_{CS} if that student already exists in the other department and the number of double majors is currently at its maximum. Secondly, since both set groupings are ordered by the students' GPA, a range restriction can be imposed. Note that S is a true subset of the two underlying set groupings (assuming that some of the double majors have GPAs of less than 3.5).

Example 3:

Let S_1 be the catalog of books of the Computer Science library, and S_2 be the catalog of books of the Mathematics library with $|S_1| \geq 1000$ and $|S_2| \geq 500$. The two libraries wish to maintain a joined catalog in order to control overlap in their respective holdings. The combined book catalog S = $S_1 \uplus S_2$ could be modeled by:

1. set cardinality: $[1300 : \infty]$

2. set repetition with parameter $=^c$: [0:15].

Interpretation: The resulting cardinality of S is constrained to be at least 1300. This implies that S_1 and S_2 together must reference at least 1300 distinct books. If that's not the case, the holdings of either or both libraries must be increased to satisfy the constraint. Similarly, this constraint would prevent the deletion of a book from either library if the combined holding were to drop below 1300. Repetition in the combined library is set to a maximum of 15. If this number is exceeded, some volumes from one or both libraries must be removed. All future insertions of new books are also subject to this constraint.

6 CONCLUSION

The concept of a semantic grouping is a major focus in semantic database modeling research. In this paper we have extended the basic set groupings and the is-a related set groupings by identifying a number of semantic restrictions for each. Combinations of these constraints produce potentially new variations of semantic groupings and provide us with higher-level mechanisms to impose semantic integrity constraints on the data. Hence, this work represents a significant step towards overcoming a major disadvantage of conventional database systems, which have to maintain constraints separately from their data and to enforce them explicitly. We would like to stress that our work proposes a framework and *not* a new model or system. This framework is targeted towards advanced applications, such as, computer-aided design, office automation, and artificial intelligence, which require the support of more sophisticated relationships among data than traditional database domains.

7 ACKNOWLEDGEMENTS

This work has been supported in part by NSF Grant CCR-8709817. We are grateful for their support.

References

[1] S. Abiteboul and R. Hull, "IFO: A Formal Semantic Database Model," *ACM Trans. on Database Systems*, Vol. 12:4, pp. 525-565, Dec. 1987.

[2] A. Borgida, "Conceptual Modeling of Information Systems", *On Knowledge Base Management Systems*, M. L. Brodie and J. Mylopoulos, (Eds.), Springer-Verlag, 1987.

[3] P. P. Chen, "The Entity-Relationship Model — Toward a Unified View of Data", *ACM Trans. on Database Systems*, Vol. 1:1, pp. 9-36, March 1976.

[4] E. F. Codd, "Extending the Database Relational Model to Capture More Meaning", *ACM Trans. on Database Systems*, Vol. 4:4, pp. 397-434, Dec. 1979.

[5] M. Hammer and D. J. McLeod, "A Framework for Data Base Semantic Integrity", *Proc. of 2nd Int. Conf. on Software Engineering*, San Francisco, CA, Oct. 1976, pp. 498–504.

[6] M. Hammer and D. J. McLeod, "Database Description with SDM: A Semantic Data Model" *ACM Trans. on Database Systems*, Vol. 6:3, pp. 351-386, Sept. 1981.

[7] R. Hull and R. King, "Semantic Database Modeling: Survey, Applications and Research Issues", *ACM Computing Surveys*, Vol. 19:3, pp. 201-260, Sept. 1987.

[8] S. N. Khoshafian and G. P. Copeland, "Object Identity", *Proc. OOPSLA'86*, ACM, pp. 406-416, Sept. 1986.

[9] W. Kim, E. Bertino and J. F. Garza, "Composite Objects Revisited", *Proc. of SIGMOD*, (SIGMOD Record Vol. 18:2, June 1989), pp. 337-347, May 31-June 2, 1989.

[10] M. Lenzerini, "Covering and Disjointness in Type Networks", *Proc. 2nd Int. Conf. on Data Engineering*, IEEE, pp. 386-393, Los Angeles, CA. Feb. 3-5, 1987.

[11] D. Maier and J. Stein, "Development and Implementation of an Object-Oriented DBMS", *Research Directions in Object-Oriented Programming*, B. Shriver and P. Wegner, (Eds.), MIT Press, 1987.

[12] E. A. Rundensteiner, L. Bic, J. Gilbert, and M.-L. Yin, "A Semantic Integrity Framework: Set Restrictions for Semantic Groupings", *IEEE Int. Conf. on Data Engineering (ICDE'91)*, Kobe, Japan, April, 9 - 11, 1991.

[13] E. A. Rundensteiner and L. Bic, "Set Operations in a Data Model Supporting Complex Objects", *Int. Conf. on Extending Database Technology (EDBT)*, Fondazione Cini, Venice, Italy, March 26–30, 1990.

[14] E. A. Rundensteiner, L. Bic, J. Gilbert, and M.-L. Yin, "Set-Related Restrictions for Semantic Groupings", Technical Report Number 89-07, Department of Information and Computer Science, University of California, Irvine, Jan. 1989.

[15] D. W. Shipman, "The Functional Data Model and the Data Language DAPLEX", *ACM Trans. on Database Systems*, Vol. 6:1, pp. 140-173, March 1981.

[16] J. M. Smith and D. C. P. Smith, "Database Abstractions: Aggregation and Generalization", *ACM Trans. on Database Systems*, Vol. 2:2, pp. 105-133, June, 1977.