Attribute Abstraction

Benkt Wangler

SYSLAB[1]
Department of Computer and Systems Sciences
The University of Stockholm and the Royal Institute of Technology
Stockholm, Sweden

Abstract

Abstractions of various kinds play a vital role in conceptual modeling and knowledge representation. These mechanisms are, however, normally applied only to object types. In this paper we define the semantics and show the usefulness of applying similar mechanisms to types of relations among objects. Specifically, we show how cardinality constraints of an abstracted attribute relate to those of its constituents. To accomplish this we employ a formalism based on information triples (binary predicates) constituting elementary assertions about a Universe of Discource.

Keywords: Conceptual modeling, abstraction, attribute abstraction

1 Introduction

We take the term 'information system' in the ISO sense of the word as described in "Concepts and Terminology for the Conceptual Schema and the Information Base" [ISO82]. This means that an information system consists of three parts, a conceptual schema, an information base and an information processor.

The information base contains assertions describing, in our case, the current state of the universe of discourse. It comprises a snapshot view of the state of affairs in the universe of discourse and contains, thus, no history. The assertions of the information base must conform to assertion types defined in the conceptual schema. A conceptual schema and information base

[1]This research was supported by the National Swedish Board for Technical Development (STU).

is totally static unless something operates on it to cause change. That something is called an information processor. Important components of the information processor are an information base management system and a conceptual schema facility.

It is widely acknowledged that the conceptual schema also plays a key role in systems analysis and design. The activity of designing a conceptual schema is usually referred to as *conceptual modeling*. According to Brodie et al. [Brodie84a]:

"conceptual modeling needs can be met only by a leap to a higher, more abstract level of system description. The required leap is analogous to the one that has been achieved in going from assembly languages to so-called high level programming languages in the tradition of Algol 60, PL/I and Lisp. The move to high level languages resulted in developing high level mechanisms for algorithm specification in terms of such concepts as variable, function, data type and control structure. The leap now needed for conceptual modeling requires high level mechanisms for the specification of large, complex models of an enterprise."

The mechanisms mentioned should support the conceptual modeling process and typically involve abstraction mechanisms like:

- *Classification*, by which objects which share common characteristics are gathered into classes. All elements of a class have the same type.
- *Generalization*, by which types with common characteristics are generalized to more general types.
- *Aggregation*, by which complex objects may be created from tuples of more elementary objects.
- *Grouping*, by which complex objects may be created from sets of more elementary objects.

As a synonym for conceptual modeling one sometimes uses the term "semantic data modeling" [Brodie84b, Hull87b]. The reason for this is the aim of conceptual models to more completely describe the universe of discourse. An important means for raising the semantical level of a conceptual model is to capture, in the conceptual schema, the various rules and laws that control the business area in question. Important classes of such rules are

1. Constraints such as mapping constraints for relations between objects e.g. 'Every person must have a name' or 'A person must not have more than one name'.
2. *Derivation rules* defining how certain information is derived from other, such as a rule defining how a derived entity of some kind is 'brought into existence', e.g. 'a *SIBLING-GROUP* entity is defined to be the set of *CHILD* entities who all have the same parents'.

Mapping constraints will be discussed in detail when studying attribute abstraction later in this paper. We will also give examples of derivation rules. Such rules are also the natural means for defining external schemata.

1.1 The Problem

The aim of this paper is to investigate how mapping constraints propagate under attribute abstractions. We take, here, the word mapping to mean a binary relation between two object sets. A tuple of such a relation normally is the manifestation of a property that one of the objects have. For example *<e1,'red'>* might mean that the entity e1 has the color red.

A mapping constraint restrict, for a certain mapping, the number of objects that may be associated with each other. These constraints are defined in the conceptual schema

- to prevent updates to the database that violate them or, in the case of deletion, possibly, propagate the delete so that the database will still be consistent with the constraints.
- to raise its semantical level in that it assures that the conceptual schema will more closely mirror the universe of discourse. This is useful when explaining to a user the meaning of the mapping or when reasoning, possibly automatically, about the propagation of these constraints, e.g. when composing or specializing mappings, as may be needed to construct external views.

Mapping constraints are in our case given as min- and max-values for the cardinality of the set of objects that may be associated to a certain object belonging to a type for which the mapping is defined [ISO82].

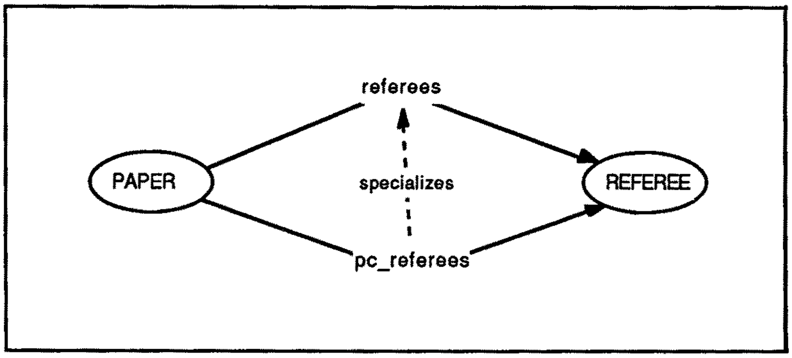

Figure 1. Attribute specialization

It is obvious that the constraints specified for a mapping will restrict what we are allowed to specify when we generalize or specialize it. Consider for example Figure 1. We say here that *pc_referees* specializes the attribute *referees* to those *REFEREE*-objects that are members of the program committee. It is then obvious that the number of pc referees of a paper can never be more than the total number of referees. If, for example, the number of *referees* is maximized to 4, so is also *pc_referees*. In other words restrictions on the *referees* attribute will constrain what is possible to specify for *pc_referees*. Also for other kinds of abstractions on attributes similar concistency rules can, as we shall see, be formulated.

Defining these *meta constraints* is an important task, since attribute abstractions are very useful, e.g. when it comes to map different data models to each other as may be needed when mapping external schemas to a conceptual schema (cf. the so called three level architecture [ISO82]) or when mapping between different data model types, as is examplified in section 4 of this paper. Formulating the relationship between mapping constraints of abstract attributes and these of their constituents is a main aim of this paper. In order to accomplish this task, we will have to clearly define the meaning of generalization, composition and aggregation of attributes. From these definitions we then derive the required properties.

1.2 Related Work

In [Hull87b], Hull and King survey the area of, what they call, semantic database modeling and briefly present 16 prominent semantic data models, all of which, in one way or the other make use of abstraction mechanisms. Aggregation and grouping are in this survey labeled "type constructors", thereby emphasizing their use as a means for constructing new complex types. A more comprehensive survey of theoretical research on constructed types may be found in [Hull87a].

Whereas object type generalization is present in most conceptual modeling languages [Hull87b], aggregation and grouping are not so commonly used. Some examples are, however, [Bracchi84] and [Schiel84]. In some of these approaches grouping is referred to as *association* [Bracchi84]. In this paper we use the term *grouping*, though, since the word association is often used to denote a binary relation in general.

The only works we have found that explicitly deal with abstracted attributes are the knowledge representation systems KL-ONE [Brachma85] and Krypton [Brachma83]. In KL-ONE, RoleSets (attributes in our terminology) belonging to specialized object types may "restrict" a RoleSet of the generic object type. This means that the subordinate RoleSet denotes a subset of the superior and that its 'values' may be restricted to some subset of the superior RoleSet's range. Also mapping constraints of the subordinate RoleSet may be restricted. A RoleSet may also be "differentiated" into several subordinate RoleSets.

Besides attribute specialization (restriction and differentiation), like in KL-ONE, Krypton [Pigman84] also supports "role-chains", i.e. composite attributes.

As can be understood from the works mentioned in this section, object type and attribute abstractions are of central importance for enhancing the expressive power of conceptual models. In the sequel of this paper we will elaborate on the meaning and use of such abstractions and perform a detailed analysis of important properties of a number of attribute abstractions.

2 Methodological Approach

To accomplish our goal of investigating the properties of abstractions, specifically the propagation of cardinality constraints when abstracting mappings, we will have to formally define the various abstraction mechanisms of interest. The basic elements of our approach are entities, properties and values. From these we build, by means of aggregating into tuples and grouping into sets, more complex objects. The most basic are elementary <entity, property, value>-assertions saying simply that for this entity, that specific property has that value. It should be noted that we do not promote our approach as a new language for conceptual modeling. It should rather be seen as a basic formalism in which it is possible to express other data model types.

2.1. Some Basic Definitions

An information base is considered to consist of objects that are related to each other via binary relations. The objects of the information base are basically of two different kinds, namely *abstract objects* (often called non-lexical objects [ISO82] or entities[2]) and *descriptor objects* (lexical objects). The names abstract object and descriptor object are due to [Lyngbaek84]. An abstract object may be elementary, represented by one single internal surrogate, or composite, constructed by means of aggregating or grouping, in one or more levels, several surrogates. In each of these cases, the single surrogate, the tuple of surrogates and the set of surrogates resp., will uniquely determine the object.

The abstract objects represent entities of the Real World, such as a person or a car, whereas descriptor objects are 'self-describing' and consist of strings of characters such as *John* or *111111-1111*.

The abstract objects are, however, of little interest unless you can see what properties they have. This is accomplished by means of named relations between objects, like e.g. e_1*--has_affiliation--*e_2 or e_1*--has_social_security_no--'111111-1111'*. Thus, a triple

<abstract-object,property-name,range-object>,

where range-object may be either an abstract object or a descriptor object, represents an assertion, i.e. a piece of information, about an entity of the Universe of Discourse. We will therefore use the word *information triple* to denote such triples. Like in CMOL [Lindenc83, Bubenko84], we do not distinguish between associations relating an entity to another entity (relationship in ER [Chen76] terminology) and properties relating an entity to a value.

An information base at a particular point in time comprises a collection of information triples. Hence, we have the following definition for an information base state:

Definition 1

An information base state, IB_n, existing at some point in time, is defined to be a non-empty set of information triples.

Every entity of the information base has a certain type as defined by the mandatory information triple *<entity,type,type_label>*, like e.g. in < e_1 , *type, 'REFEREE'*>. The type labels are taken from a value set containing names of the types defined in the conceptual schema.

Now, let *e* denote an abstract object occurring in IB_n. Then we define *a(e)* to be the set of

[2]In the rest of this paper we will alternate between the terms entity and abstract object.

'target-objects' associated via the 'property' a to the entity e, and correspondingly for the inverse relation. Formally we get

Definition 2

Suppose e is an entity of IB_n. Then

$a(e)= \{r: <e,a,r> \in IB_n\}$

and

$a^{-1}(r)= \{e: <e,a,r> \in IB_n\}$

However, since we want to be able to discuss composite attributes, like $a(b(e))$, we also define $a(X)$, where X is an arbitrary set of entities occurring in IB_n, we also define:

Definition 3

$$a(X) = \bigcup_{e \in X} a(e)$$

and

$$a^{-1}(Y) = \bigcup_{r \in Y} a^{-1}(r)$$

where X is defined as above and Y is a set of arbitrary objects (descriptor or abstract) occurring in IB_n.

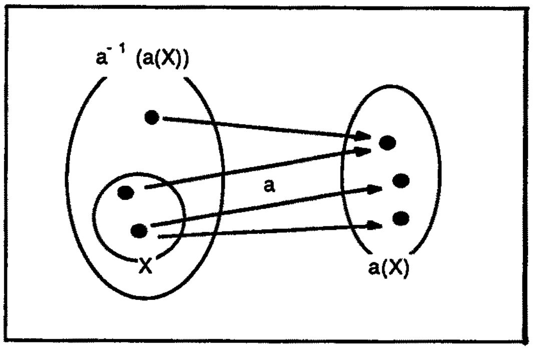

Figure 2. An example showing X, $a(X)$ (the image of X under a) and $a^{-1}(a(X))$

It follows immediately from definition 3 that

2-1 $a(\emptyset) = \emptyset$ and $a^{-1}(\emptyset) = \emptyset$.

The set of objects associated to a set of entities X via a, $a(X)$, is called the image of X under a (see Figure 2). For example *affiliation(e)* refers to the set of *UNIVERSITY* entities to which the *PERSON* entity e is affiliated, whereas ssn^{-1}*('111111-1111')* refers to the single-element set containing the abstract object representing the person having this particular social security number.

To be exact, the function symbol, a, of definitions 2 and 3 is overloaded, since it represents functions of different types ('object $\rightarrow$ set' and 'set $\rightarrow$ set', resp.). However, we will not go into a detailed discussion of this here, but only consider the function defined in definition 2 to be a special case of the function defined in definition 3. In other words it is a (set $\rightarrow$ set)-function, the argument of which is a singleton set. This means that, given that E is the total set of entities occurring in IB_n and R is the total set of objects of IB_n, a could also be described as a function

$$a : 2^E \rightarrow 2^R$$

We may also refer to $a(E)$ as the target or codomain of a and $a^{-1}(R)$ as the domain of a. Note also that $a^{-1}(a(X))$ is, in general, not equal to X (see Figure 2).

2.2 The Conceptual Schema

Having defined the concept of an information base, we now proceed to define the conceptual schema of the information base. We will, however, not do that formally in this paper, but only briefly explain how we look upon these matters.

The conceptual schema is basically a collection of entity types, data types, action types, constraint specifications and derivation rules to which the information base, in any state, has to conform. It contains, thus, a set of entity types, defining the entities allowed to occur in the information base and a set of data types, defining the permissible values. It is presupposed that the schema is not changed during the life time of the information system.

An entity type is defined to be a set of property specifications, called attributes, telling

- the name of a property, a, and
- the range object type (entity type or data type)

A data type comprises a definition of a set of descriptor objects. Data types are defined in terms of certain pre-defined value classes, such as INTEGER, REAL, CHARACTER, STRING etc. A descriptor object of IB_n is said to be of a certain data type if it belongs to the value class constituting the data type. The information base is at no time allowed to contain descriptor objects other than those being of types defined in the schema.

Action types correspond to types of events that may occur in the universe of discourse and are the means for causing change in the information base. Since, in this paper, events are not of primary interest, we will not go further into defining the details of action types here.

We will now proceed to define the notion of mapping constraints for attributes. We do that formally, since it is of major importance henceforth

Definition 4

The mapping constraint, a_{constr}, for the attribute a, mapping from entities of type A to objects of type R, is defined to be the tuple $(a_{min} : a_{max}, a^{-1}_{min} : a^{-1}_{max})$, such that for all e being of type A and all r being of type R:

a_{min} is the greatest integer, such that $/a(e)/ \geq a_{min}$ holds in every information base state,

a_{max} is the smallest integer, such that $/a(e)/ \leq a_{max}$ holds in every information base state,

a^{-1}_{min} is the greatest integer, such that $/a^{-1}(r)/ \geq a^{-1}_{min}$ holds in every information base state,

a^{-1}_{max} is the smallest integer, such that $/a^{-1}(r)/ \leq a^{-1}_{max}$ holds in every information base state.

Here $a_{min}, a^{-1}_{min} \in N$[3] $and\ a_{max}, a^{-1}_{max} \in (N - \{0\}) \cup \{*\}$, where the asterisk indicates that the cardinality may be any number $\in N$, greater than or equal to a_{min} or a^{-1}_{min} resp. Since * could mean any number ≥ 0 we say that $n \leq *$ and $n \cdot *$ [4]$= *$ for any $n \in N$.

If $a_{min} \geq 1$, we say that a is total, otherwise it is partial.

The set of entities in IB_n being of type A, is said to be the extension of A and is denoted by δA (notation due to [Sowa84]). In other words

$\delta A = \{ e: <e,type,'A'> \in IB_n \}$.

Correspondingly, the relation

$\delta a = \{<e,r>: <e,a,r> \in IB_n\}$

is called the extension of the attribute a..

Updates to the information base are handled under the control of the information base management system which ensures that what is specified in the conceptual schema is not violated. Provided consistency rules like those formulated in 3-1, 3-3 and 3-5 are included in the conceptual schema facility, it will also have the possibility to initially check the schema for consistency.

2.2.1 Graphical Notation

Figure 3 describes a subset of the well-known 'IFIP Working Conference example' [Olle82], that will be used for examplifying various matters in the sequel. This application concerns an information system, supporting the arrangement of an IFIP working conference. Familiarity with the example will be assumed.

[3] N denotes, throughout this paper, the set of positive integers (incl 0).

[4] $=$ *n times* * , as is needed in various formulas in the sequel.

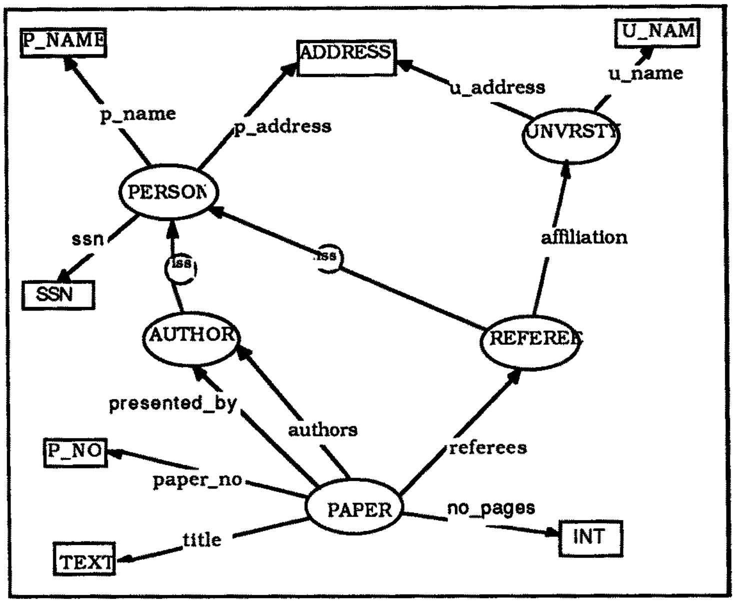

Figure 3. A conceptual schema for part of the 'IFIP case'

We use a graphical description technique for the schema, where ellipses represent abstract object types, rectangles represent descriptor types and attributes are represented by arrows. The abstraction mechanisms of generalization, aggregation and grouping are, as is examplified in figures 3 and 4, graphically represented by a circle inscribing the symbols *iss* (is subset), *x* (cartesian product) and *P* (powerset) resp.

2.3 Abstractions on Object Types

In general terms the meaning of the word *abstraction* is to disregard detail in order to gain a more general view of some situation. This can be done in many different ways.

Firstly one may construct a more general object type by overlooking distinguishing properties and only see to what is common to a number of other object types. This is called *generalization* and an example may be the object type *PERSON* of Figure 3, which is a generalization of *AUTHOR* and *REFEREE*. The abstraction of generalization establishes an *is-a* relationship between objects.

Secondly one may overlook the internal structure of an object, i.e. one wants to talk about a thing as a whole without bothering about its parts. However, an object may be built up by constituents in at least two different ways. It may consist of exactly one each of a number of parts, whence we call the composite object an *aggregate*, or it may consist of an unknown number of exactly one kind, whence we call it a *group*. Examples are here the aggregate *couple* consisting of one man and one woman and the group object *program committee* which is a group of persons. The abstraction of aggregation establishes a *part-of* relationship between objects, whereas grouping establishes a *member-of* relationship. We will henceforth use the words *component* to denote the parts of an aggregate and *member* to denote the members of a group object.

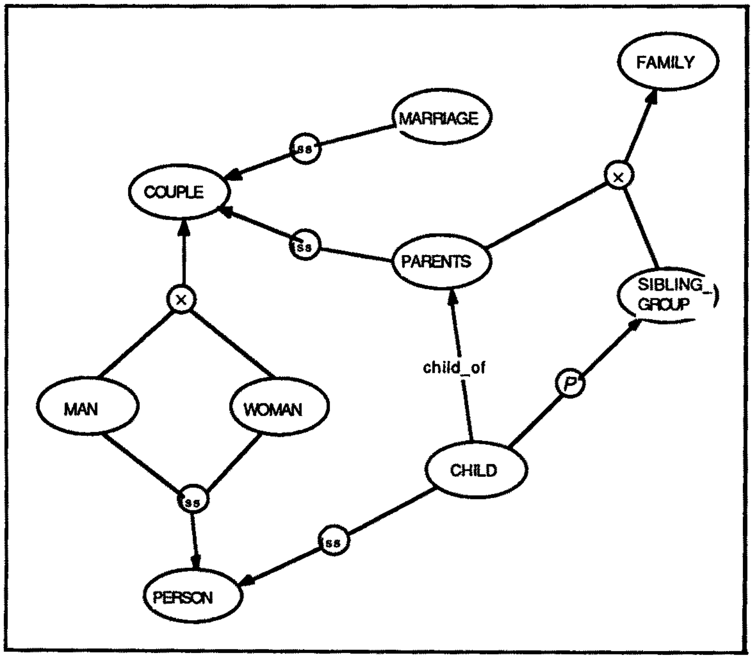

Figure 4. Abstracted object types

Aggregation and grouping may, like the entity type *FAMILY* in Figure 4, be applied to aggregates or groups of objects, thus creating more and more complex objects. In some situations an object type could also be grouped or aggregated into several aggregate or group types. This would constitute something similar to multiple inheritance.

3 Attribute Abstractions

In this section we will define and discuss three kinds of abstractions on attributes, namely generalization, composition and aggregation. For the sake of simplicity we define composition and aggregation for only two component attributes. The definitions and rules given below could, however, easily be increased to an arbitrary number of attributes. We also present a number of propositions defining the relationship between mapping constraints of abstracted attributes and those of their components. The proofs of the most important of these propositions are given in the appendix.

3.1 Generalized Attributes

It has been mentioned already in the introduction of this work that *pc_referees* of Figure 1 is a specialization of the attribute *referees*. This is the same as saying that *referees* is a generalization of *pc_referees* and the exact semantics is that the set of pc referees related to a specific paper, *e* say, must always be a subset (possibly the empty set) of the total set of referees for that paper or, more formally, that *pc_referees(e)* is always a subset of *referees(e)*.

Due to this definition it is, for the example of Figure 1, immediately clear that the number of referees related to a certain paper via the *pc_referees* attribute cannot exceed the number of objects related via *referees*. In other words $pc\_referees_{max}$ cannot exceed $referees_{max}$. By the same reason $pc\_referees_{min}$ cannot exceed $referees_{min}$ and correspondingly for the inverse attribute. Suppose for example that $referees_{min} = n\ (n \geq 0)$. Since this means that the number of referees for a paper may be as low as *n* and since the number of pc_referees cannot be greater than the number of referees, it is obvious that we cannot prescribe any value for $pc\_referees_{min}$ that is greater than *n*.

For another example, consider Figure 3. Here we could state that the attribute *presented_by* is a specialization of *authors*. This would entail the constraint that the person who presents a paper must be one of the authors of that paper.

In these cases a main reason for having the specialized attributes of *pc_referees* and *presented_by* in the schema is to give us a possibility to pick up exactly those referees of the paper who are also members of the program committee and the one of the authors who is going to present the paper.

There are also other reasons for specializing attributes. In Figure 1 the attribute *pc_referees* is the only way membership of the program committee is represented. Suppose instead we have a specialized object type *PC_MEMBER*, as in Figure 5. The range of the *pc_referees* attribute would then be restricted to *PC_MEMBER* and thus be derivable according to:

$$pc\_referees(e) = \{x: x \in \delta PC\_MEMBER \;\&\; referees(e) = x\};$$

As for Figure 1 it is clear that $pc\_referees_{max}$ cannot exceed $referees_{max}$. Neither can $pc\_referees_{min}$ exceed $referees_{min}$ nor $pc\_referees^{-1}{}_{max}$ exceed $referees^{-1}{}_{max}$.

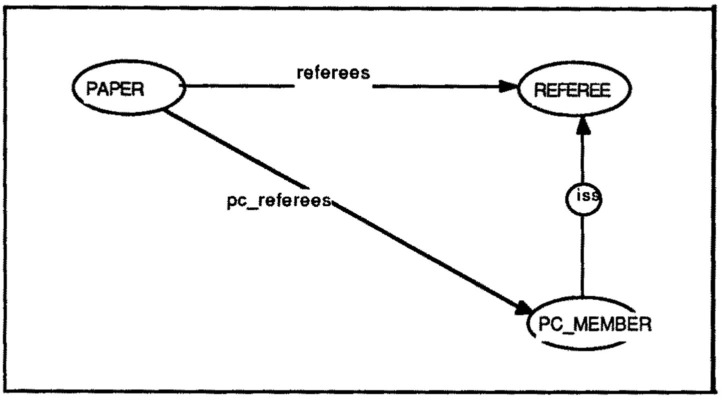

Figure 5. Specialized attribute with specialized range

Since, in the definition of minimum constraints on attribute inverses (definition 4) we require that only objects of the range object type are considered, the case for $pc\_referees^{-1}{}_{min}$ is a bit more complicated, though. Suppose for example that $referees^{-1}{}_{min} = 1$, i.e. each referee must review at least one paper. Then there may very well also be a rule saying that a pc member must review at least two papers, i.e. $pc\_referees^{-1}{}_{min} = 2 \geq referees^{-1}{}_{min}$. On the other hand, there may be other situations, where the semantics of the specialized attribute is not at all tied to the specialized range object type and, hence, the minimum constraint for the attribute inverse may be less than the corresponding constraint for the generic attribute. Thus, we cannot say anything about the relationship between $pc\_referees^{-1}{}_{min}$ and $referees^{-1}{}_{min}$.

In Figure 6 we have a specific category of papers, called *X_PAPERS*, which we demand to be reviewed by at least 4 referees, while for normal papers it is sufficient with 3 referees. We could represent this rule by defining an attribute *x_referees*, for which we could specify $x\_referees_{min} = 4$, whereas $referees_{min} = 3$. It is obvious that the x_referees of an x_paper should be equal to the referees of that paper and that the set of x_papers reviewed by a referee must be a subset of the total set of papers reviewed by that referee. Hence *x_referees* should be defined as a specialization of *referees*.

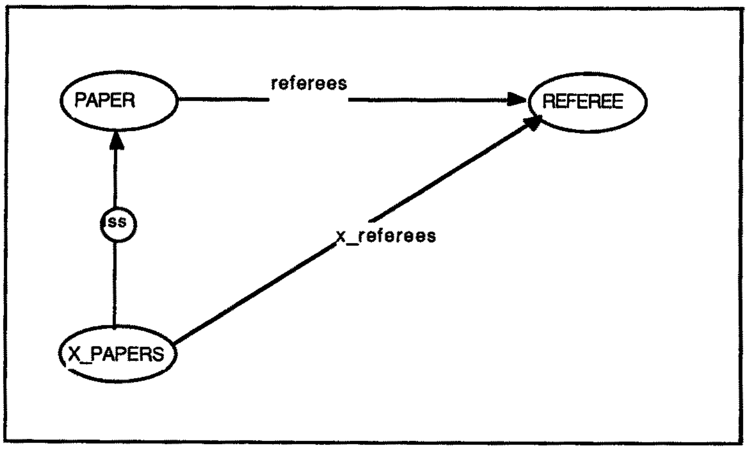

Figure 6. Specialized attribute with specialized domain

The case we have here is symmetrical to Figure 5 in that we cannot say anything in general about the relationship between the minimum constraints of the generalized and specialized attributes (now taken in the positive direction).

In Figure 7 both the domain and range of *x_pc_referees* are specialized and hence we cannot know anything about the relationships between the minimum constraints of *x_pc_referees* as related to those of *referees*. For the maximum constraints, on the other hand, it is as always clear that the values for the generic attribute is always greater than the corresponding values for the specialized attribute. This is due to the fact that in the definition (definition 4) of the maximum constraints it does not matter from where the arguments are taken.

Note that it is possible to have specialized attributes that are not derivable as those of figures 5, 6 and 7 are. *pc_referees* of Figure 1 is an example of this. Non-derivability is also the case when the specific semantics of the specialized attribute is not in any way related to its specialized domain and range.

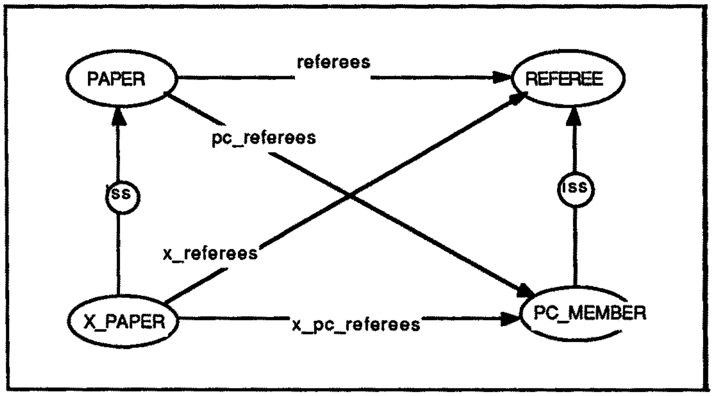

Figure 7. Specialized attribute with specialized domain and range (x_pc_referees)

We may summarize our findings concerning attribute specialization by establishing that minimum and maximum constraints for specialized attributes are always less than the corresponding values for the generic attribute, provided the domain and range object types are the same. If they are not, we do not know anything about the relationship between minimum constraints, while what is said about maximum constraints still hold.

We now proceed to formally defining the notion of attribute generalization.

Definition 5

If a_g is an attribute relating entities of type E_g to objects of type R_g and a_s is an attribute relating entities of type E_s to objects of type R_s, where E_g is a generalization of E_s and R_g is a generalization of R_s, then a_g is said to be a generalization of a_s iff in any database state, IB_n

$a_g(e) \supseteq a_s(e)$ and

$a_g^{-1}(r) \supseteq a_s^{-1}(r)$, for any e and r.

If a_g is a generalization of a_s we also say that a_s is a specialization of a_g.

The relationships between mapping constraints discussed above may now be summarized as follows,

3-1 If the attribute a_g, relating entities of type E_g to objects of type R_g is a generalization of the attribute a_s, relating entities of type E_s to entities of type R_s, then

1) $a_{gmin} \geq a_{smin}$, provided $E_g = E_s$

2) $a_{gmax} \geq a_{smax}$

3) $a_g{}^{-1}{}_{min} \geq a_s{}^{-1}{}_{min}$, provided $R_g = R_s$

4) $a_g{}^{-1}{}_{max} \geq a_s{}^{-1}{}_{max}$.

For an example, suppose for Figure 1, that $referees_{constr} = (3{:}4, 1{:}5)$. This means that each paper must be reviewed by between 3 and 4 referees and that each referee must review between 1 and 5 papers. According to our findings, by stating that $pc\_referees_{constr} = (2{:}4, 0{:}4)$, it would be possible to specify that every paper must be reviewed by at least 2 pc members and that each pc member may not review more than 4 papers. However, it is not possible to state that each pc member must review at least one paper. This is due to the fact that we have no independent way to distinguish which referees are pc members. For the schemata of figures 8 and 10 it would be permitted, though, to state $pc\_referees_{constr} = (2{:}4, 1{:}4)$.

This section is about generalization. In spite of this we have mostly been discussing specialization. However, regardless of whether the specific or generic attributes are primary, the same conditions regarding mapping constraint propagation hold.

3.2 Composite Attributes

Sometimes we have the need to construct composite attributes. Consider for example Figure 8 and suppose that the object types in the schema have lots of other attributes than those shown in the figure. The secretary returning reviewers reports to the author would probably not be interested in all the details about persons. Instead we ought to define an external schema giving access only to the name and address of the persons who are going to present each paper. This could be accomplished by means of the composite attributes *presented_by.p_name* and *presented_by.p_address*, as shown in Figure 3. The meaning of these composite attributes should be intuitively clear.

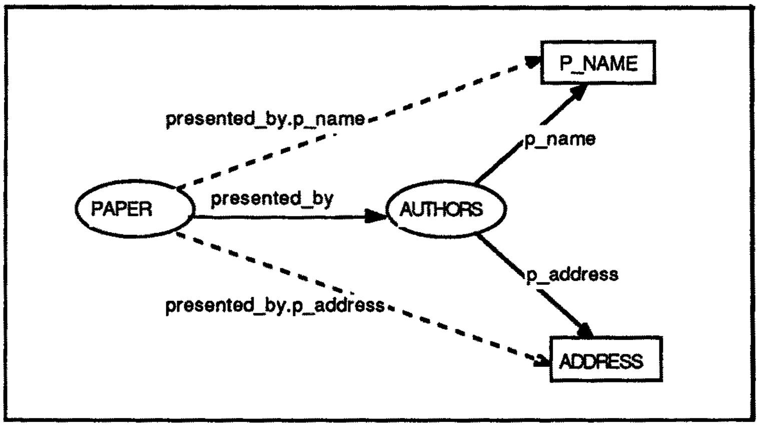

Figure 8. Composite attributes

We now proceed to define the concept of composite attribute formally.

Definition 6

If a is an attribute relating entities of type A to entities of type B and b is an attribute of B, then $a.b$ is said to be the composite attribute of A, such that

$a.b(X) = b(a(X))$ for any X, such that $\delta A \supseteq X$ and the inverse

$(a.b)^{-1}(Y) = a^{-1}(b^{-1}(Y))$ for any Y, such that $\delta B \supseteq Y$.

Again note that $(a.b)^{-1}(a.b(X)) \neq X$, in general.

We may now fomulate the relationship between the mapping constraints of a composite attribute and those of its component attributes as follows:

3-2 If a is an attribute relating A objects to B objects and b is an attribute relating B objects to C objects, then

1) $a.b_{min} \geq min(a_{min} \cdot b_{min}, b_{min})$

2) $a.b_{max} \leq a_{max} \cdot b_{max}$

3) $(a.b)^{-1}_{min} \geq min(b^{-1}_{min} \cdot a^{-1}_{min}, a^{-1}_{min})$

4) $(a.b)^{-1}_{max} \leq a^{-1}_{max} \cdot b^{-1}_{max}$.

Regarding 1) and 3) above it is impossible to say anything stronger than this since e.g. in the case of 1), although there may be many elements in *a(e)* they might all be associated to the same *C*-object.

3-2 is a meta constraint, restricting what constraints are possible to specify for a composite attribute given certain constraints for the components and vice versa. It should be perfectly clear, though, that 3-2 gives only limits. Only by knowing the application could the exact values be told.

As an example consider again Figure 3 and suppose the mapping constraints for the attribute *referees* are *(3:3,...)*. This means that each paper must be reviewed by exactly three referees. If nothing more is said they might all be employed at the same university. Given that $affiliation_{min} = 1$ they may never be employed at less than one university, though, since $referees_{min}$ (= 3) is greater than 0.

However, it might very well be, that there is a rule saying that the referees of a paper must all be employed at different universities. This rule could now be taken into account by specifying the constraint, *(3:3,...)* for the composite attribute *referees.affiliation.*

A specific use of composition is to create a direct reference from a key descriptor object to some other object type via one or more abstract object types, like e.g. $paper\_no^{-1}.referees('123')$. Suppose also that *paper_no* is the key of *PAPER*, i.e. $paper\_no_{constr} = (1:1,1:1)$. Then, since *PAPER* could be one-to-one replaced by *PAPER_NO*, obviously

$(paper\_no^{-1}.referees)_{constr} = referees_{constr}$.

Hence we have the following

3-3 Suppose a_1 and a_2 are two attributes of A, then

if $a_{1constr} = (1{:}1,1{:}1)$ then $(a_1^{-1}.a_2)_{constr} = a_{2constr}$.

Finally, there may also be situations where the maximum constraint of a composite attribute may be zero. Considering Figure 3, suppose, just for the sake of example, that the authors of a paper, must not be affiliated to a university (also supposing, for a moment, that $affiliation_{min} = affiliation^{-1}{}_{min} = 0$). Now, this constraint could be represented by specifying $(authors.\ affiliation)_{constr} = (0{:}0,0{:}0)$.

3.3 Aggregate Attributes

Often there is a need to specify aggregates of attributes. This is, for example, the case when more than one attribute is needed to identify an object. Consider, e.g., the entity type *PERSON* of Figure 9. Here the attributes of *p_name* and *p_address* in combination uniquely identify a person. This implies that, although there may be many persons having the same name and many persons having the same address, there must not be two persons having the same name and address combination. This, in turn, indicates that the mapping constraints for the *<p_name, p_address>* aggregate are more restrictive than the constraints of the component attributes suggest.

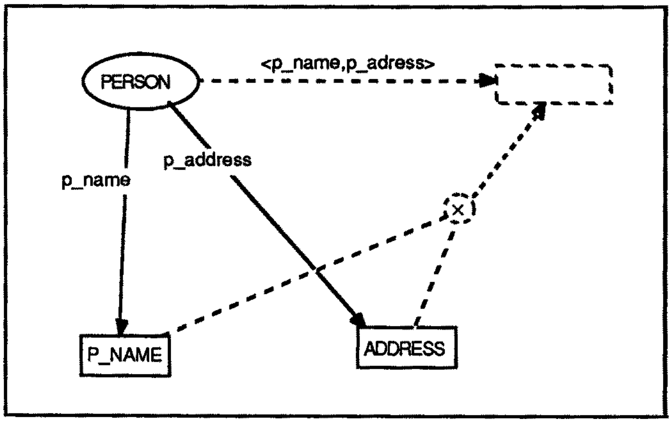

Figure 9. Aggregation of attributes p_name and p_address

To analyze this further we now proceed to define the notion of attribute aggregate.

Definition 7

If the entity type A has the attributes a_1 and a_2 relating A to B_1 and B_2 resp (B_1 and B_2 need not be disjoint) then $<a_1, a_2>$ is said to be an aggregate attribute relating objects of type A to tuples of $\delta B_1 \times \delta B_2$, such that, for any information base state, IB_n,

$<a_1, a_2>(e) = \{<r_1, r_2>: <e, a_1, r_1> \in IB_n \,\&\, <e, a_2, r_2> \in IB_n\}$

and the inverse is

$<a1, a2>^{-1}(<r1, r2>) = \{ e: <e, a_1, r_1> \in IB_n \& <e, a_2, r_2> \in IB_n\}.$

Like before we also say

$$<a_1, a_2>(X) = \bigcup_{e \in X} <a_1, a_2>(e)$$

and

$$<a_1, a_2>^{-1}(Z) = \bigcup_{<r_1, r_2> \in Z} <a_1, a_2>^{-1}(<r_1, r_2>)$$

where $\delta A \supseteq X$ and $\delta B_1 \times \delta B_2 \supseteq Z$.

Once again note that $<a_1, a_2>^{-1}(<a_1, a_2>(X)) \neq X$ in general.

Returning to our discussion of Figure 9, definition 7 now implies that $<p\_name, p\_address>^{-1}(<\text{'Clark Kent'}, \text{'Metropolis'}>)$ refers to the single-element set containing the *PERSON*-entity having this particular *p_name* and *p_address*.

However, definitions 7 and 2 also give us the following:

3-4 If e is an entity of some type and $a1$ and $a2$ are attributes of that entity type then

if $a_1(e) = \emptyset$ or $a_2(e) = \emptyset$ then $<a_1, a_2>(e) = \emptyset$.

We now proceed to formulate the mapping constraints of $<a_1, a_2>$ and $<a_1, a_2>^{-1}$ as related to those of a_1 and a_2.

3-5 If a_1 and a_2 are attributes relating A objects to B_1 objects and B_2 objects respectively, then, for any information base state, IB_n,:

1) $<a_1, a_2>_{min} \geq a_{1min} \cdot a_{2min}$

2) $<a_1, a_2>_{max} \leq a_{1max} \cdot a_{2max}$

3) $<a_1, a_2>^{-1}{}_{min} \geq 0$

4) $<a_1, a_2>^{-1}{}_{max} \leq min(a_1{}^{-1}{}_{max}, a_2{}^{-1}{}_{max})$

Again we have a consistency rule restricting what would be legal to specify for an attribute constructed by means of abstraction, given certain constraints for the components. Consider, again, the example of Figure 3 and suppose $p\_name_{constr} = (1{:}1, 1{:}\, *)$ and $p\_address_{constr} = (1{:}1, 0 : *)$ ($p\_address^{-1}{}_{min} = 0$ since *p_address* shares its codomain with *u_address*). By means of 3-5 we can now immediately conclude that

a) $<p\_name,p\_address>_{min} = <p\_name,p\_address>_{max} = 1$;

This is trivial.

b) $<p\_name,p\_address>^{-1}_{min} \geq 0$;

$<p\_name,p\_address>^{-1}$ relates tuples of $\delta NAME \times \delta ADDRESS$ to objects of type $PERSON$. However, there must, of course be lots of invalid *p_name-p_address* combinations, i.e. combinations for which there are no corresponding persons. Hence $<p\_name,p\_address>^{-1}_{min} = 0$. This would have been the case even if $p\_address^{-1}_{min} = 1$. In fact, it is, most likely, the case for most aggregate attributes. It is probably only for quite degenerate cases one could think of a value greater than *0*.

c) $<p\_name,p\_address>^{-1}_{max} \leq *$.

It was, however, said earlier that *p_name* and *p_address* together uniquely identify a person. The semantics of this is that $<p\_name,p\_address>^{-1}_{max} = 1$.

Thus we have found that for this example $<p\_name,p\_address>_{constr} = (1{:}1,0{:}1)$.

Also the case of mutually exclusive attributes may be taken care of by means of constraining aggregate attributes. Supposing we live in a society where no one is allowed to have both a cat and a dog and supposing ownership of these kinds of pets is represented by two (partial) attributes *dog* and *cat*, we could represent the mentioned rule by $<cat,dog>_{constr} = <0{:}0,0{:}0>$.

4 Mapping Entities to Relations

In section 3 we have demonstrated the possibility to express important database constraints by means of constraining the mappings for abstracted attributes. In this section we want to point out how abstracted attributes may be utilized for defining mappings between the conceptual schema and the relational model [Codd70]. To make it possible to express other than binary relations, our results need to be generalized to include aggregation and composition of an arbitrary number of attributes as well as nesting aggregation and composition. We will, however not deal explicitly with these questions here.

Since the relational model only allows descriptors (values), the problem of transforming a conceptual schema to the relational model is one of replacing abstract objects with expressions involving only descriptor objects.

As it turns out, aggregating the attributes of entities is a natural means for mapping our data model to the relational model. Attributes which do not relate to descriptor objects may by means of composition, be made to do so.

As a first example, consider once again Figure 3. Here, the expression

$<ssn,p\_name,p\_address>(\delta PERSON)$

would mean a set of $<ssn,p\_name,p\_address>$ triples describing persons in the information base. It is, thus, a (ternary) relation containing only descriptors, i.e. the extension of a 'relational model relation'. In the usual formalism of the relational model [Date86] this relation (intension) would be written *PERSON(ssn,p_name,p_address)*. Hence, we can define our transformation by equating this with the aggregate attribute expression, i.e.

$$PERSON(ssn,p\_name,p\_address) = <ssn,p\_name,p\_address>(\delta PERSON)$$

Given that the attributes of the left side are identified with those on the right side in turn, a set of such equations define a mapping to the relational model.

By taking $\delta PERSON$ as the argument of the aggregate attribute, we express that we want the aggregate attribute tuples for all persons. Attributes that are partial would, however, according to 3-4, result in the corresponding tuples being left out. Thus, we will always get relations without null-values. Suppose, for a moment, that *p_address* above is partial, i.e. $p\_address_{min} = 0$. This means that our relation would contain only those persons who have addresses. To deal with partial attributes we have, therefore to employ a strategy where we always treat such attributes separately. Given that *ssn* and *p_name* are total, the relation $<ssn,p\_name>\ (\delta PERSON)$ would describe all persons in the database, whereas $<ssn,p\_address>\ (\delta PERSON)$ would list only those who have addresses.

The expression

$$<ssn,p\_name>(\ \delta AUTHOR)$$

denotes a subset of $<ssn,p\_name>(\ \delta PERSON)$ describing all the authors. Hence, we have here a means for expressing specialized relations simply by giving an expression for the requested type as an argument to the aggregate attribute.

The second subtype of *PERSON* would yield the relation

$$<ssn,p\_name,p\_address,affiliation.u\_name,affiliation.u\_address>(\ \delta REFEREE)$$

where *affiliation* is made refer to descriptors by composing it with *u_name*. Note that we have here a relation containing all the attributes that *REFEREE* inherits from *PERSON*.

The primary key of these relations would be *ssn*, since it uniquely determines a *PERSON*-entity, which in turn uniquely determines a *P_NAME* and a *P_ADDRESS*. Hence we have a (indirect) functional dependency from *ssn* to *p_name* and *p_address*. $<p\_name,p\_address>$ might be an alternate key. For the transformation of entities to tuples to be one-to-one, each entity type must have a unique key, otherwise there could be identical tuples for certain entities. These duplicate tuples would then be 'collapsed' to a single tuple, due to the set property of relations.

Given, now, that *p_address* is total, the mapping constraint of our relation is $<ssn,p\_name, p\_address>_{constr} = (1{:}1,0{:}1)$, since each entity of type *PERSON* corresponds to exactly one triple and since there obviously are triples in $\delta SSN \times \delta P\_NAME \times \delta ADDRESS$ that do not correspond to any person, whereas each triple corresponds to at most one person. However, by taking $<ssn,p\_name,p\_address>(\ \delta PERSON)$ we select exactly those triples corresponding to existing persons.

Since, in these examples, all attributes are directly and fully functionally dependent on the entity key, the mentioned relations are all fourth normal form. This is not always the case, though, since for instance some attributes of an entity type may be multivalued. An example of this is the entity type *PAPER*, where the attributes *authors* and *referees* are multivalued and, hence, should each be taken separately together with the key to yield 4NF relations, i.e.

<paper_no,authors>(δPAPER) and *<paper_no,referees>(δPAPER).*

We have in these examples considered only transformation to relations in the classical relational model sense. It seems, however, reasonable that similar mechanisms may also be used for transforming conceptual schemata according to our modeling approach to other data models, such as non first normalform relations [Makinou77], the network model or the hierarchical model.

5 Concluding Remarks

In this paper we have considered abstractions, specifically on attributes. Using a formalism based on elementary, three-place, assertions we build a theory in which important properties of abstracted attributes are derived. In particular, we consider the propagation of mapping constraints under attribute abstraction.

Attributes may be abstracted by

- generalizing one or more attributes to form a generic attribute subsuming the former attribute(s). More specifically, this means that the extension of the generic attribute is a superset of the extension of the subordinate attribute(s). This entails, for the simplest case that the mapping constraints of the generic attribute must be greater than those of the specific attributes.

- composition of two or more attributes to form a 'chain' attribute directly relating some object set to objects to which it is, originally, only indirectly related. It is shown that the mapping constraints of a composite attribute is not always the product of those of its components. Composite attribute mappings may also be used to express important constraints that do not follow directly from the mapping constraints of the component attributes. An important use of composite attributes is to make abstract objects referable by descriptors.

- aggregating several attributes defined over the same entity type. This is needed e.g. when defining keys for attributes that do not have a single attribute key. Moreover, mapping constraints for aggregate attributes may be used to express other important constraints, i.e. where the mapping constraints of the aggregate attribute is more restrictive than the constraints of the component attributes suggest. This is useful, for example, in the case where an aggregate attribute is a candidate key for some entity type.

It should be noted that composite and aggregate attributes are always derivable, i.e. the way they are defined, uniquely determine their extensions. Generalized attributes, on the other hand, are not derivable. Generalization over attributes should, instead, simply be considered a constraint demanding certain attribute extensions to be subsets of others.

In section 4 we have indicated the way in which abstractions may be used to define 'relational model relations' for abstract objects.

Acknowledgements

The author is grateful to Janis Bubenko and Paul Johannesson for carefully reading and commenting on an earlier version this paper. Thanks also to Moira Norrie for giving valuable suggestions both regarding the contents and the language. Thanks also to Erik Knudsen who read and commented an early draft of the paper.

REFERENCES

ANSI75 ANSI/X3/SPARC, "Study Group on Data Base Management Systems: Interim Report 75-02-08", in ACM SIGMOD Newsletter, FDT, Vol. 7, No. 2, 1975.

Bracchi84. G. Bracchi and B. Pernici, "SOS: A conceptual model for office information systems," Proceedings of ACM SIGMOD Database Week Conference, pp. 108-116, San Jose, Calif., May 1984.

Brachma83 R.J. Brachman, R.E. Fikes and H.J. Levesque, " Krypton: A Functional Approach to Knowledge Representation" in IEEE Computer, pp 67-73, October 1983.

Brachma85 R.J. Brachman and J.G. Schmolze, " An overview of the KL-ONE Knowledge Representation System", Cognitive Science, Vol.9, No. 2, April-June, 1985.

Brodie84a. M.L. Brodie, J. Mylopoulos, J.W. Schmidt (Eds.), "On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases and Programming Languages", Springer Verlag, New York, 1984.

Brodie84b. M.L. Brodie, "On the Development of Data Models", in "On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases and Programming Languages", pp 19-47, Springer Verlag, New York, 1984.

Bubenko84 J. Bubenko and E. Lindencrona, Konceptuell Modellering, Informations-analys, Studentlitteratur, Lund, Sweden, 1984. (In Swedish)

Cardelli85 L. Cardelli, P. Wegner, "On Understanding Types, Data Abstraction, and Polymorphism", in ACM Computing Surveys, Vol. 17, No. 4, pp. 471-522, December 1985.

Chen76. P.P.S. Chen, "The Entity-relationship Model - Towards a Unified View of Data," ACM Transactions on Database Systems, vol. 1, no. 1, pp. 9-36, 1976.

Codd70 E.F. Codd, "A Relational Model of Data for Large Shared Data Banks.", CACM 13, No.6 (June 1970). Republished in *Milestones of Research - Selected Papers 1958-1982: CACM 25th Aniversary Issue, CACM* 26, No. 1, January 1983.

Date86 C.J. Date, "An Introduction to Database Systems, Volume I", Fourth Edition, Addison-Wesley Systems Programming Series, Addison-Wesley Publishing Company, 1986.

Hull87a. R. Hull, " A Survey of Theoretical Research on Typed Complex Database Objects", in Databases, J Paredaens (Ed), Academic Press, London, 1987.

Hull87b. R. Hull and R. King, "Semantic Database Modeling: Survey, Applications and Research Issues", ACM Computing Surveys, vol 19, no 3, pp 201-260, 1987.

ISO82. ISO, "Concepts and Terminology for the Conceptual Schema and the Information Base," Report N695, ISO/TC9/SC5/WG3, 1982. Eds J.J. van Griethuysen

Lindenc83 E. Lindencrona-Ohlin, J.A. Bubenko jr, "Towards a Formal Syntax for a Data Modeling Language - DMOL", SYSLAB Working Paper no. 63 version 2, Department for Information Processing and Computer Science, University of Stockholm, S-106 91 Stockholm, December 1983.

Lyngbaek84 P. Lyngbaek, D. McLeod, "Object Management in Distributed Information Systems", in ACM Transactions on Office Information Systems, Vol. 2, No. 2, pp. 96-122, April 1984.

Makinou77 A. Makinouchi, "A Consideration on Normal Form of Not-Necessarily-Normalized Relations in the Relational Data Model", in Proceedings of the Conference on Very Large Data Bases (Tokyo, 1977), pp. 48-69, 1977.

Olle82. T W Olle, H G Sol, and A A Verrijn-Stuart (Editors), Information System Design Methodologies: a Comparative Review, North Holland, Amsterdam, 1982.

Pigman84. V. Pigman, "Krypton: Description of an Implemantation, Volume 1", Artificial Laboratory, Schlumberger Palo Alto Research, Report no 40, Palo Alto, November 1984.

Schiel84 U. Schiel, A.L. Furtado, E.J. Neuhold, M.A. Casanova, "Towards Multi-level and Modular Conceptual Schema Specifications", in Information Systems, Vol. 9, No. 1, pp. 43-57, 1984.

Sowa84 J.F. Sowa, "Conceptual Structures - Information Processing in Mind and Machine", Addison-Wesley, 1984.

Wangl89 B. Wangler, "On the Use of Abstractions in Database Modeling: Propagation of Mapping Constraints under Attribute Abstraction", SYSLAB Report No. 61, Department for Computer and Systems Sciences, Stockholm University, January 1989.

Appendix

In this appendix we present the most important propositions of our paper together with proofs.

Generalized Attributes

3-1 If the attribute a_g, relating entities of type E_g to objects of type R_g is a generalization of the attribute a_s, relating entities of type E_s to entities of type R_s, then

1) $a_{gmin} \geq a_{smin}$, provided $E_g = E_s$

2) $a_{gmax} \geq a_{smax}$

3) $a_g{}^{-1}{}_{min} \geq a_s{}^{-1}{}_{min}$, provided $R_g = R_s$

4) $a_g{}^{-1}{}_{max} \geq a_s{}^{-1}{}_{max}$.

Proof: 3-1 follows immediately from definitions 4 and 5 and the fact that

$a_g(e) \supseteq a_s(e)$ implies that $/a_g(e)/ \geq /a_s(e)/$ and the fact that

$a_g{}^{-1}(e) \supseteq a_s{}^{-1}(e)$ implies that $/a_g{}^{-1}(e)/ \geq /a_s{}^{-1}(e)/$.

Composite Attributes

In order to arrive at how the mapping constraints of $a.b$ relate to those of a and b (3-2), we first give the following:

3-2a If a is an attribute relating objects of type A to objects of type B and

$\delta A \supseteq X \neq \varnothing$, then

$a_{min} \leq /a(X)/ \leq /X/ \cdot a_{max}$.

Proof:

1) We first prove the left part:

$a_{min} \leq /a(X)/$ follows immediately from the definition of $a(X)$ and the fact that X is a non-empty subset of δA (otherwise $/a(X)/ = 0$ by (2-1), regardless of a_{min}).

2) For the right part we have from definition 3

$$|a(X)| = |\bigcup_{e \in X} a(e)| \le \sum_{e \in X} |a(e)|^{5} \le |X| \cdot a_{max}$$

3-2 If a is an attribute relating A objects to B objects and b is an attribute relating B objects to C objects, then

1) $a.b_{min} \ge min(a_{min} \cdot b_{min}, b_{min})$

2) $a.b_{max} \le a_{max} \cdot b_{max}$

3) $(a.b)^{-1}{}_{min} \ge min(b^{-1}{}_{min} \cdot a^{-1}{}_{min}, a^{-1}{}_{min})$

4) $(a.b)^{-1}{}_{max} \le a^{-1}{}_{max} \cdot b^{-1}{}_{max}$.

Proof:

We prove the formulas one at a time.

1) If $a_{min} = 0$ formula 1) simply says that $a.b_{min} \ge 0$, which is trivial (in fact it is, of course $= 0$, for this case). If $a_{min} > 0$ it says that $a.b_{min} \ge b_{min}$, which holds according to the following:

Suppose $e \in \delta A$. Then according to 3-2a and the fact that $a_{min} > 0$ (i.e. $a(e) \ne \emptyset$):

$|a.b(e)| = |b(a(e))| \ge b_{min}$;

2) Suppose e $\in \delta A$. Then according to definition 6 and 3-2a

$|a.b(e)| = |b(a(e))| \le |a(e)| \cdot b_{max} \le a_{max} \cdot b_{max}$

3-4) are proved accordingly.

Aggregate Attributes

We now proceed to formulate the mapping constraints of $<a_1, a_2>$ and $<a_1, a_2>^{-1}$ as related to those of a_1 and a_2. For the proof we need, however, the following:

3-5a Suppose $P(x)$ and $Q(x)$ are two predicates. Then

$|\{<x,y>: P(x) \& Q(y)\}| = |\{x: P(x)\}| \cdot |\{x: Q(x)\}|$.

[5] $|A \cup B| \le |A| + |B|$

Proof:

$P(x)\&Q(y)$ holds for each tuple $<x,y>$, where $x \in \{x:P(x)\}$ and $y \in \{x:Q(x)\}$.

Hence $|\{<x,y>:P(x)\&Q(y)\}| = |\{x:P(x)\}| \cdot |\{x:Q(x)\}|$.

3-5 If a_1 and a_2 are attributes relating A objects to B_1 objects and B_2 objects respectively, then, for any information base state, IB_n,:

1) $<a_1 , a_2>_{min} \geq a_{1min} \cdot a_{2min}$

2) $<a_1 , a_2>_{max} \leq a_{1max} \cdot a_{2max}$

3) $<a_1 , a_2>^{-1}{}_{min} \geq 0$

4) $<a_1 , a_2>^{-1}{}_{max} \leq min(a_1{}^{-1}{}_{max} , a_2{}^{-1}{}_{max})$

Proof:

We first prove 1) and 2). From definition 7 and 3-5afollows

$|<a_1 , a_2>(e)|=|\{<r_1 , r_2>:<e , a_1 , r_1> \in IB_n \& <e , a_2 , r_2> \in IB_n\}| =$

$|\{r_1 :<e , a_1 , r_1> \in IB_n\}| \cdot |\{r_2:<e , a_2 , r_2> \in IB_n\}| =$

$|a_1(e)| \cdot |a_2(e)|$

Now, 1) and 2) follows from definition 4.

3) is trivial. See discussion following this proof.

For 4) we have from definitions 6 and 2

$|<a_1 , a_2>^{-1}(<r_1 , r_2>)| =$

$|\{e:<e , a_1 , r_1> \in IB_n\&<e , a_2 , r_2> \in IB_n\}|=$

$|\{e:<e , a_1 , r_1> \in IB_n\} \cap e:<e , a_2 , r_2> \in IB_n\}| \leq$

$min(|\{e:<e , a_1 , r_1> \in IB_n\}|,|\{e:<e , a_2 , r_2> \in IB_n\}|)^6 =$

$min(|a_1{}^{-1} (r_1)|,|a_2{}^{-1}(r_2)|) \leq min(a_1{}^{-1}{}_{max},a_2{}^{-1}{}_{max})$.

6 $|P \cap Q| \leq min (|P|,|Q|))$