

# An OODB “Part” Relationship Model

Michael Halper, James Geller, and Yehoshua Perl  
Institute for Integrated Systems, CIS Department  
and  
Center for Manufacturing Systems  
New Jersey Institute of Technology  
Newark, NJ 07102 USA

## Abstract

The whole-part organizational method is one which pervades human thinking, and as such plays an important role in data modeling. This is especially true in manufacturing, design, graphics, multi-media, and document processing—areas targeted by object-oriented databases (OODBs). In this paper, we formally define OODB relationships which provide constraints and functionalities that enforce real-world, whole-part interactions between objects of the database. These OODB “part” relationships capture a variety of semantics including exclusiveness/sharing, single-/multi-valuedness, cardinality range-restriction, ordering of definite or indefinite number, essentiality, dependency, and value propagation. For each kind of relationship, we present a graphical symbol which enhances an OODB graphical schema representation that we have previously developed. Among the part relationships presented, two of them, the global exclusive and the class exclusive, refine the notion of exclusiveness between wholes and parts. Two others allow dependency to be directed from the part to the whole, or vice versa. Similarly, data values may be propagated either upward or downward along the part relationship. A basis for the realization of the part relationships in the context of an existing OODB data model is discussed.

## 1 Introduction

The whole-part organizational method is one which pervades human thinking [4, 22]. As such, it plays an important role in data modeling, where it is often referred to as aggregation or, conversely, part decomposition [18, 21]. It is particularly important in the construction of conceptual models in advanced domains such as manufacturing [14], design [16], graphics [10],

multi-media [23], and document processing [13]. Since all these areas have been the focus of object-oriented databases (OODBs) [1, 2, 6], it is not surprising that some such systems (e.g., [15, 16, 17, 20, 23]) have supported a notion of “part.”

In earlier work [8, 10], we investigated different kinds of part relationships as they relate to graphical deep knowledge [9]. We have also devised a realization for a variety of part relationships in the context of an existing OODB data model [11]. In this paper, we expand our previous work by supplying formal definitions and graphical symbols for a diverse group of part relationships, some of which were presented in [11]. The graphical notation enhances the OODB graphical schema representation we introduced in [12].

The part relationships included in our model have the following characteristics: exclusiveness/sharing, single-/multi-valuedness, cardinality range-restriction, ordering of definite or indefinite number, essentiality, dependency, and value propagation. Our notion of exclusiveness extends previous work (e.g., [15]) through refinement into two kinds, global exclusiveness and class exclusiveness. Dependency in our model may be specified from the part to the whole, or vice versa. Similarly, value propagation may be directed upward or downward along the part relationship.

The remainder of the paper is organized as follows. In the next section, we present basic terminology and definitions. After that, in Section 3, we formally define the different part relationships and present graphical symbols for each. Section 4 discusses the problem of combining part relationships. Section 5 contains the conclusions.

## 2 Terminology

In this section, we present some terminology and notation which will be used throughout the paper. Fol-

lowing [22], we will occasionally refer to a “part” object as a *meronym*. A whole object will be called an integral object or *holonym*. We will sometimes refer to the class of a part as the meronymic class; likewise, the class of the integral object will be referred to as the holonymic class. For example, if there is a part relationship defined between classes `engine` and `car` and engine  $e$  is part of car  $c$ , then  $e$  is a meronym and  $c$  is a holonym. The classes, `engine` and `car`, are the meronymic and holonymic classes, respectively. The “part” relationship itself will be referred to as the *meronymic* relationship.

As we mentioned above, we have defined a graphical schema representation for OODBs, the details of which can be found in [12]. Here, we will be employing the following conventions. A class is represented as a box enclosing the name of the class (Figure 1). An attribute is represented by an ellipse (surrounding a name) which is attached to a class via an unlabeled line (Figure 16). The subclass relationship is a thick arrow directed from the subclass to the superclass (Figure 3). An “ordinary” relationship is denoted by a (thin) labeled arrow pointing from the source class to the target class (Figure 2).

The set  $E(C)$  will denote the extension of the class  $C$ , i.e., the set of all instances of the class.

### 3 The Part Relationship

In this section, we describe the details of the different part relationships included in our model. For each one, we give a formal definition and a graphical symbol to be used in the context of the OODB graphical schema representation. Before getting to the details of each individual relationship, we first give the definition of a “generic” part relationship from which all the others will be derived.

In our model, we define the generic part relationship between a meronymic class  $B$  and holonymic class  $A$  as a relation  $P$  from  $E(B)$  to  $E(A)$ . The pair  $(b, a) \in P$  indicates that an instance  $b$  of class  $B$  is part of instance  $a$  of class  $A$ . The graphical symbol used to represent the part relationship between classes  $B$  and  $A$  is a thick, broken line connecting the two classes (Figure 1).

We have tried to make our part relationship symbols intuitive by exploiting the mnemonic value of the graphical icon. The mnemonic device for these symbols is the association between the pieces of the line and the parts of the object. The thickness is used to highlight the part relationship in the overall context

of the schema, and particularly to contrast it to “ordinary,” user-defined relationships (represented as thin lines). It is essential that the “part” hierarchy stand out visually if one is to gain an intuitive understanding of the application.

We encourage the placement of the holonymic class spatially above the meronymic class in the schema diagram (as in Figure 1). Following [20], we place a diamond head on the end of the line touching the holonymic class. We have chosen this solution for two reasons. First, it maintains consistency with an already existing notation [20]. Also, the use of the diamond head avoids the implication of directedness associated with arrows. This is desirable given that our part relationship is a two-way access and constraint mechanism.

In order for the part relationship to truly capture “part” semantics, it must be refined with one or more of the following:

- additional constraints which impose a real-world, whole-part interaction between the objects of the participating classes (e.g., the exclusiveness constraint, discussed below).
- additional functionalities such as dependency and value propagation which enhance overall database performance and modeling capabilities.
- query support for whole-part access and retrieval.

In the remainder of this section, we describe the constraints and functionalities which lead to a number of different part relationships with characteristics such as the following: exclusiveness/sharing, single-/multi-valuedness, cardinality range-restriction, ordering of definite or indefinite number, and essentiality. There are also two part relationships which express dependency, and two others which permit value propagation.

Before continuing, let us note some further notational conventions. To maintain consistency with our graphical notation, we will write  $b \diamond a$  instead of  $(b, a) \in P$  to indicate that the object  $b$  is part of the object  $a$  with respect to a part relationship whose underlying relation is  $P$ . In fact, instead of  $P$ , we will use the symbol  $\diamond$  (possibly with a subscript) to stand for the relation itself.  $B \diamond A$  will denote the part relationship between meronymic class  $B$  and holonymic class  $A$ .

To define the constraints imposed by the part relationships, we will need the following. Assume that  $B \diamond A$ .

**Definition 1**  $\forall a \in E(A)$ , let  $M_{\diamond}(a) = \{b \mid b \in E(B) \wedge$

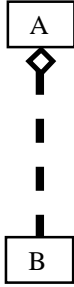


Figure 1: Generic part relationship

$b \diamond a$ .  $M_\diamond(a)$  will be referred to as the *meronym set* of  $a$  with respect to  $\diamond$ .

**Definition 2**  $\forall b \in E(B)$ , let  $H_\diamond(b) = \{a \mid a \in E(A) \wedge b \diamond a\}$ .  $H_\diamond(b)$  will be referred to as the *holonym set* of  $b$  with respect to  $\diamond$ .

Our realization of a part relationship is based on its automatic expansion into a class of its own [11]. For example, the schema of Figure 1 is expanded automatically by the system into the schema of Figure 2. The part relationship, in particular, is transformed into a class B-PART-A which is connected to classes  $A$  and  $B$  via the (ordinary) relationships *holonym* and *meronym*, respectively. Likewise,  $A$  and  $B$  are connected to B-PART-A by the relationships  $p$  and  $w$ . The instances of B-PART-A are “relationship objects” which represent actual “part” connections between instances of  $A$  and  $B$ . (That is, they are objects which represent elements of the relation  $\diamond$ .)

The first advantage to our approach is that the realization does not require the introduction of any extraordinary new OODB modeling constructs. Second, the arrangement provides the means for traversal from the whole to the part, and vice versa, and offers a convenient way of performing value propagation (see below). Finally, as objects in their own right, the instances of the part relationship class can, if necessary, be endowed with attributes in a similar fashion to relationships in the ER [3] and other semantic data models [18].

### 3.1 Exclusive and Shared Part Relationships

Part relationships in general can be divided along the lines of *exclusive* and *shared* [15, 17]. An *exclusive part relationship* enforces the restriction that a

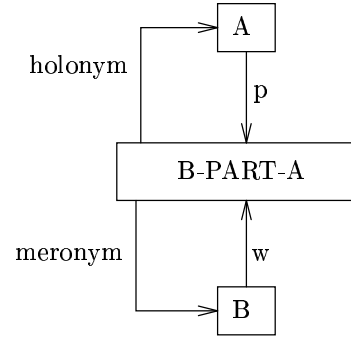


Figure 2: Realization of Figure 1

given meronym can be a component of only a single holonym. In other words, the holonym is the sole owner of the meronym. Of all the part relationships we will introduce, the exclusive relationship is perhaps the most intuitive because part modeling is most often associated with physical assemblies such as cars, bridges, and buildings, things that one can “go out and kick” [4]. For such items, the exclusiveness restriction is quite natural: Two cars cannot share the same engine.

While no two cars can share an engine, it is also the case that a car and, say, an airplane cannot share one either. Therefore, the exclusive part relationship between the classes *engine* and *car* must have ramifications for the entire database topology, restricting not only “part” references from cars to engines but from objects of other classes to engines as well. There are times, however, when we would like to confine the exclusive reference restriction to a single holonymic class. Consider a computer science publication database which contains scholarly journals and books (and, in particular, books which are compilations of articles). An excerpt from the conceptual model of this database is shown pictorially in Figure 3, where we use the generic part relationship symbol to indicate that class *article* is in a part relationship with both *journal* and *compilation* (the latter being a subclass of *book*). Ordinarily, different journals do not contain the same article. Therefore, it is sensible to impose this constraint on the database. However, an article can appear as part of some compilation (a common practice in the area), and so we do not want the exclusiveness constraint between *article* and *journal* to have any implications on the relationship between *article* and *compilation*.

For this reason, we distinguish between two types of exclusiveness, *global exclusiveness* and *class exclu-*

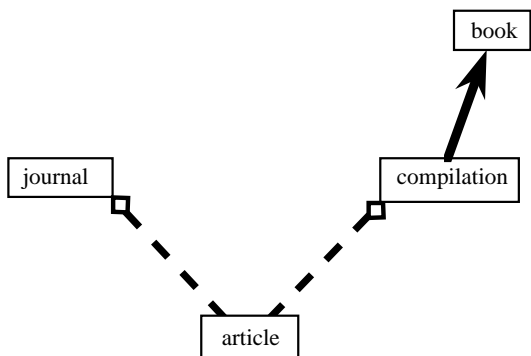


Figure 3: Excerpt from publication schema

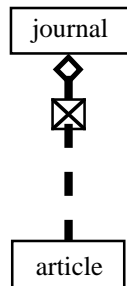


Figure 4: Revised excerpt

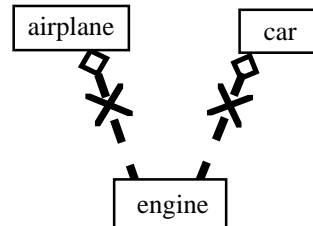


Figure 5: Exclusive relationships

*siveness*. An exclusive part relationship, such as the one between `engine` and `car`, which affects the entire database topology will be referred to as a *global exclusive part relationship*. This kind can be found in a number of existing systems (e.g., [15, 17]), where it is simply called the exclusive part relationship. We too will usually drop “global” and just call it exclusive. The *class exclusive part relationship* is one which only enforces the exclusiveness constraint on the relationship between the participating classes, as between `article` and `journal`. Both the exclusive and class exclusive relationships will be given formal definitions and their own graphical representations below.

Part relationships which are not exclusive are called *shared* [15]. A shared part relationship puts no restrictions on the number of holonyms that a given meronym can be part of, allowing the meronym to be freely shared. The part relationship between `article` and `compilation` in the example discussed above is shared. The same article can be included in any number of compilations.

To formalize the definitions of these three part relationships, we place different constraints on the cardinality of the holonym sets of the parts. It is interesting to note that the resulting theoretical ordering with respect to the extent of the constraints (namely, shared, followed by class exclusive, and then exclusive) is exactly opposite to the “intuitive” ordering that we used to introduce these part relationships.

**Definition 3**  $B \diamond A$  is a *shared part relationship* iff there exist no constraints on  $|H_\diamond(b)|$ ,  $\forall b \in E(B)$ .

In particular, the cardinality of such sets must be allowed to be greater than one. Because of this lack of constraints, we use the generic graphical symbol to represent the shared relationship. (Actually, our generic symbol denotes a single-valued, shared part

relationship. Sharing may also be combined with multi-valuedness which is discussed in the next section.) We refer once again to Figure 3 for the example of the shared part relationship between `article` and `compilation`.

**Definition 4**  $B \diamond A$  is a *class exclusive part relationship* iff  $\forall b \in E(B)$ ,  $|H_\diamond(b)| \leq 1$ . In other words, it is class exclusive iff  $\diamond$  is a partial function from  $E(B)$  to  $E(A)$ .

To express the fact that a part relationship is class exclusive, we add a rectangle enclosing an “X” directly behind the diamond head of the generic symbol. The “X” is derived from “eXclusive.” The rectangle, our symbol for a class, brings to mind “class.” An example of the class exclusive part relationship can be seen in the revised excerpt from the publication database schema shown in Figure 4.

The (*global*) *exclusive part relationship* between a pair of classes  $B$  and  $A$  is equivalent to a class exclusive relationship if it is the only one that  $B$  participates in. Therefore, assume that  $B$  is in part relationships  $\diamond_1, \diamond_2, \dots, \diamond_n$  with classes  $A_1, A_2, \dots, A_n$ , respectively.

**Definition 5** The part relationship  $\diamond_i$  is *global exclusive* (exclusive, for short) iff  $\forall b \in E(B)$ , if  $\exists a \in E(A_i)$  such that  $b \diamond_i a$ , then  $\forall j \neq i$ ,  $|H_{\diamond_j}(b)| = 0$ .

In other words, if an instance of  $B$  is part of an instance of  $A_i$ , then it cannot be part of any other object in the database. Thus, the part relationship  $\diamond_i$  affects the entire database topology; in particular, it puts constraints on all the other part relationships that  $B$  participates in as the meronymic class.

The graphical symbol used to represent the exclusive relationship is the generic symbol with an “X” placed behind the diamond head (Figure 5). Here, the

“X” should be read as the global eXclusiveness constraint.

### 3.2 Single-/multi-valued Part Relationships

The holonyms in a part relationship may have a single part from the meronymic class or they may have many. To accommodate these situations, we introduce a number of single-/multi-valued part relationships, defined in terms of constraints on the  $M_\diamond$ 's.

**Definition 6**  $B \diamond A$  is a *single-valued part relationship* iff  $\forall a \in E(A), |M_\diamond(a)| \leq 1$ . In other words, the relationship is single-valued iff the converse relation  $\diamond^{-1}$  (i.e., the “has-part” direction) is a partial function from  $E(A)$  to  $E(B)$ .

The generic part symbol aptly expresses the single-valuedness of this part relationship as it is a single-lined connection (Figure 1). The mnemonic here is “single line equals single part.” This is in contrast to the multi-valued part symbol where a dual line is employed to convey multiplicity (Figure 6). The multi-valued relationship is defined presently.

**Definition 7**  $B \diamond A$  is a *multi-valued part relationship* iff there exist no constraints on  $|M_\diamond(a)|, \forall a \in E(A)$ .

We note that according to our definitions the characteristics of exclusive/sharing and single-/multi-valuedness are completely independent of each other and can be freely mixed and matched to form such part relationships as the *single-valued, shared; single-valued, class exclusive; multi-valued, exclusive*; etc. (In fact, as we noted earlier, the generic symbol in Figure 1 is actually the single-valued, shared part relationship.) Because of this orthogonality, we demonstrated the graphical symbols for the exclusive/shared variations without any regard to single-/multi-valuedness. Likewise, in this section, we will illustrate the graphical symbols without regard to exclusiveness/sharing.

**Definition 8**  $B \diamond A$  is a *range-restricted part relationship* iff  $\forall a \in E(A), m \leq |M_\diamond(a)| \leq n$ , where  $m$  and  $n$  are integers with  $0 \leq m \leq n$ .

Pictorially, the range-restriction is shown as a numerical range alongside the dual-lined symbol of the multi-valued part relationship. Note that even though we are using parentheses, the range is interpreted to include both endpoints. As an example, see Figure 7 where we show the (partial) model of trucks with two

to nine axes. The upper or lower bounds of this part relationship may be omitted for an “m or greater” or “0 to n” interpretation. Graphically, a dash replaces the omitted bound.

**Definition 9** The *fixed-cardinality part relationship* is a range-restricted part relationship with  $m = n$ . The value of  $m$  is said to be the *multiplicity* of the relationship.

If only engines with six cylinders are of interest in the database, then a fixed-cardinality relationship of multiplicity six would be used to model this. The notation for this relationship is similar to that for the range-restricted, except that the upper and lower bounds are consolidated into a single number (Figure 8).

**Definition 10** The *essential part relationship* is the fixed-cardinality relationship of multiplicity 1.

If we wish to require that all cars in our database have an existent engine, we would employ an essential part relationship to model this. Since the essential relationship is not actually multi-valued, we drop the dual line from its symbol. We also forgo the “1” in parentheses and instead place a circle directly on the broken line (Figure 8). This representation follows the convention we introduced in [12] where we indicated the essentiality of attributes and “ordinary” relationships using a circle. (This notation is also consistent with that used in [23].)

**Definition 11** The *multi-valued, essential part relationship* is a range-restricted part relationship with lower bound 1 and no upper bound.

We also use the circle for this relationship, but here we maintain the dual line. Figure 9 contains an example which states that an article must have at least one section.

### 3.3 Ordered Part Relationships

In many modeling situations, we find different objects of a meronymic class functioning in different capacities within an integral object. Consider, for example, a document processing database where we have the classes `memo` and `text_block`, representing respectively memoranda and blocks of “raw” text. A memo can be defined as the composition of a header, body, and trailer, each of which can be viewed as text blocks. Therefore, it might seem feasible to model `memo` with

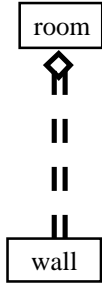


Figure 6: Multi-valued part relationship

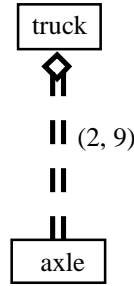


Figure 7: Trucks with 2–9 axles

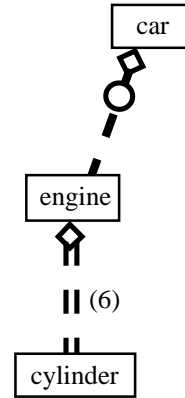


Figure 8: Car part hierarchy

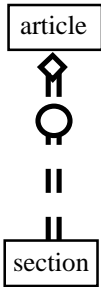


Figure 9: Multi-valued, essential

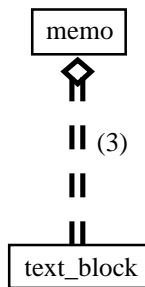


Figure 10: Inadequate model for memo

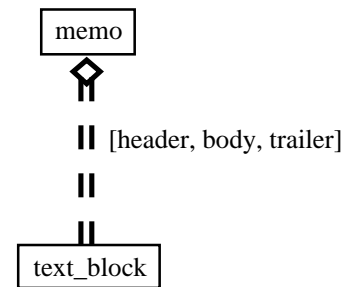


Figure 11: A better model

a multi-valued part relationship as in Figure 10. However, this model fails to preserve any distinction between the parts of a memo. The header cannot be distinguished from the body, nor can it be distinguished from the trailer.

To correctly model such a situation, what is needed is the ability to explicitly name the parts or give them an ordering. For this purpose, we define the *ordered part relationship of definite number* as follows.

**Definition 12** Let  $(s_1, s_2, \dots, s_n)$  be an  $n$ -tuple of selectors with  $n > 1$ . The *ordered part relationship of definite number*  $B \diamond A$  is an  $n$ -tuple of relations  $(\diamond^{(s_1)}, \diamond^{(s_2)}, \dots, \diamond^{(s_n)})$  from  $E(B)$  to  $E(A)$  such that the converse of each relation is a partial function from  $E(A)$  to  $E(B)$ . If  $b \diamond^{(s_i)} a$ , then  $b$  is said to be part of  $a$  in the capacity of  $s_i$ .

The condition in the definition requiring the converses to be partial functions guarantees that a holonym will have at most one part functioning in any given capacity. It will be recalled that the definitions of the

holonym and meronym sets require the existence of a single relation  $\diamond$ . Here, however, the definition provides an  $n$ -tuple of relations. To remedy this, we define  $\diamond = \bigcup_{1 \leq i \leq n} \diamond^{(s_i)}$  for ordered part relationships. Note that  $b \diamond a$  indicates that  $b$  is a part of  $a$  independent of capacity.

We graphically represent the ordered part relationship by placing the selectors in square brackets adjacent to the representation of the multi-valued part relationship. The dual line is retained because holonyms in this case still have many parts, each in different capacities. As an example, see Figure 11 where we show a better model for memo.

There is an interesting subtlety that arises when the ordering of parts is combined with exclusiveness/sharing. Exclusiveness constraints affect the way meronyms are shared by different holonyms. In other words, the constraints specify how the different part relationships of a given meronymic class interact. As defined, the ordered part relationship can be viewed as a collection of  $n$  separate part relationships between a

pair of classes. So the question arises, how should these be allowed to interact? For example, while we might want to permit the sharing of a header among many memos, we might not want to allow a text block to be both the header and body of the same memo. To accomplish the latter, we would need to impose some kind of exclusiveness restriction between the relations corresponding to `header` and `body`. We are currently studying this kind of *intra-object* exclusiveness and will report on the results in a future paper.

The *ordered part relationship of indefinite number* is used when the parts from the meronymic class form a sequence whose length cannot be determined *a priori*. For example, books comprise an ordered sequence of chapters; however, the number of chapters varies from book to book and cannot be fixed when the class `book` is defined.

**Definition 13** The *ordered part relationship of indefinite number*  $B \diamond A$  is a sequence of relations  $\{\diamond^{(i)}\}$  from  $E(B)$  to  $E(A)$  such that  $\forall i$  the converse of  $\diamond^{(i)}$  is a partial function from  $E(A)$  to  $E(B)$ .

If  $b \diamond^{(i)} a$ , we say that  $b$  is part of  $a$  in the capacity of its *i*th part (with respect to the given relationship). For instance, a chapter could be part of a book as its fourth chapter. As above, we define  $\diamond = \bigcup_i \diamond^{(i)}$ . Note that the definition of this relationship does not force any continuity on the sequence of parts associated with a given holonym. For example, while it is being written, a book may have a chapter four but no chapter three. A continuous version of this relationship requiring the existence of parts 1 through  $i - 1$  given the existence of part  $i$  can easily be defined. Such a relationship would apply to completed books.

The graphical symbol for this relationship is similar to that for the other ordered relationship. Here, though, we replace the list of selectors with an ellipsis to indicate the indefinite length of the sequence (Figure 12). There may be times when we need a label for the sequence of parts in order to make the model more readable. For example, assume that instead of `chapter`, we have the class `text_block`, whose instances are to be used as chapters for books. Then, we would prepend the label “chapter” to the square brackets to make the interpretation of the part relationship clearer (Figure 13).

### 3.4 Dependent Part Relationships

*Dependency* semantics are often desired when modeling with parts. For example, if a large integral object, such as a CAD drawing, is deleted, we may want

all its parts deleted automatically so as to avoid having to search them out and delete them manually. In our model, we define two kinds of dependency, *part-to-whole dependency* as in the previous example and *whole-to-part dependency*. The first of these is defined as follows.

**Definition 14**  $B \diamond A$  is *part-to-whole dependent* iff  $\forall a \in E(A)$ , if  $a$  is deleted, then  $\forall b \in E(B)$  such that  $b \diamond a$ ,  $b$  is deleted.

To express part-to-whole dependency in our notation, an arrowhead pointing in the direction of the holonymic class is placed immediately behind the diamond head of the generic symbol. In general, the arrowhead points to the class whose objects trigger the deletion of other objects. An example of the dependency of `block` on `engine` is shown in Figure 14.

**Definition 15**  $B \diamond A$  is *whole-to-part dependent* iff  $\forall b \in E(B)$ , if  $b$  is deleted, then  $\forall a \in E(A)$  such that  $b \diamond a$ ,  $a$  is deleted.

We express this dependency graphically by facing the arrow toward the meronymic class, as in Figure 15, where we see that a bicycle is deleted automatically when its frame is.

### 3.5 Value Propagating Part Relationships

We now define two part relationships which support upward and downward value propagation [7, 17]. Value propagation refers to the flow of a data *value* across the part relationship. As a modeling tool, it is useful for expressing certain functional dependencies between integral objects and their parts. As an example, a car may be modeled such that its age is equal to the age of its frame. In other words, the attribute *age* of class `car` would be defined to be identical to the attribute *age* of class `frame`, which is a meronymic class in relation to `car`. In such a case, instead of storing the value of *age* at both classes, the value should be stored at `frame` and propagated upward through the part relationship to `car` as needed. In this way, *age* need not be stored multiple times, and its value is guaranteed to be the same at both `car` and `frame`.

In the discussion below, we follow [24] in defining the properties of a class [namely, its attributes, relationships, and (reader) methods] as functions which map instances of the class into values of an associated type. For example, if the attribute *name* is defined for class `person`, then *name* is a function which maps instances of `person` into “nameType” (e.g., “string”).

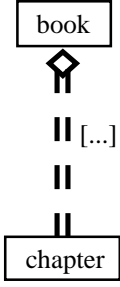


Figure 12: Indef. number of chapters

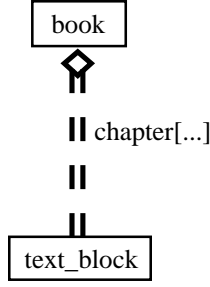


Figure 13: Text blocks as chapters

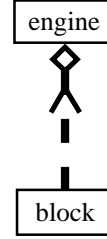


Figure 14: Part-to-whole dependency

**Definition 16** Let  $\pi : E(B) \rightarrow \pi$ -type be a property of  $B$ . The *upward propagating part relationship*  $B \diamond A$  which propagates the value of  $\pi$  is a pair  $(\diamond, F_\pi)$ , where  $\diamond$  is a relation from  $E(B)$  to  $E(A)$  whose converse is a partial function, and  $F_\pi : E(A) \rightarrow \pi$ -type is defined as follows:

$$F_\pi(a) = \begin{cases} \pi(b), & \text{if } \exists b \in E(B) \text{ such that } b \diamond a \\ \text{undefined,} & \text{otherwise} \end{cases}$$

The requirement that  $\diamond$  have a converse which is a partial function insures that  $F_\pi$  is well-defined. In a realization of this relationship, the public interface of the class  $A$  would be augmented with a message to retrieve the value of  $F_\pi$  for a given instance. This message would be identical to the message used to retrieve property  $\pi$  from an instance of  $B$ . As an illustration, consider the above example where *age* is propagated from *frame* to *car*. The interface of *car* would be augmented with a message “age” which for any instance of *car* would retrieve the value of the function  $F_{\text{age}}$  defined as

$$F_{\text{age}}(c) = \begin{cases} \text{age}(r), & \text{if } \exists r \in E(\text{frame}) \text{ s.t. } r \diamond c \\ \text{undefined,} & \text{otherwise} \end{cases}$$

The upward propagating part relationship is represented graphically by placing the name of the property being propagated in parentheses alongside the generic symbol. An upward-pointing arrowhead is written in front of the parentheses to indicate the direction of the propagation (Figure 16).

The value propagation mechanism could be defined such that all the properties of the meronymic class are made available to the holonymic class. We have chosen to concentrate on a single property because the propagation of all properties is ordinarily not meaningful in the context of a part relationship. A holonym does not normally require many of its part’s properties. We can, of course, extend the definition to a set of properties.

The *downward propagating part relationship* is used in the case where a data value of the whole determines something about its parts. For example, in the real world, if a filing cabinet is composed of steel, then its drawers are probably composed of steel, too. In general, we could opt to model drawers such that they are always composed of the same material as their cabinets. We stress that within our part model, such an arrangement would *not* represent a default (see, e.g., [19]), but rather a definitive modeling decision requiring all drawers to obtain their material make-up from their filing cabinets.

The definition of the downward propagating relationship is analogous to that of its upward propagating counterpart. Due to lack of space, we omit the actual specification. We do point out that the graphical symbol used is identical to that for the upward propagation except that the prepended arrowhead points downward (Figure 17).

## 4 Combining Part Relationships

The part relationships that have been presented above can be combined in various ways to yield relationships such as *shared*, *range-restricted*; *class exclusive*, *essential*; *exclusive*, *fixed-cardinality*; and so on. Due to space limitations, we will not elaborate further on valid combinations. A complete exposition will appear in a future paper. We will, however, point out that certain combinations of part relationships within a schema lead to inconsistencies, particularly when dependency is used together with range-restrictions.

**Theorem 1** *There exist inconsistent part schemata.*

**Proof** Consider the schema in Figure 18, where  $B$  is in a part-to-whole dependent relationship  $\diamond_1$  with  $A$  and an essential relationship  $\diamond_2$  with  $C$ . Assume that  $b \diamond_1 a$  and  $b \diamond_2 c$ . What happens if we try to delete

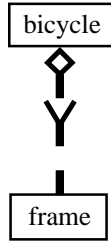


Figure 15: Whole-to-part dependency

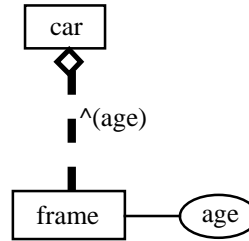


Figure 16: Upward propagation

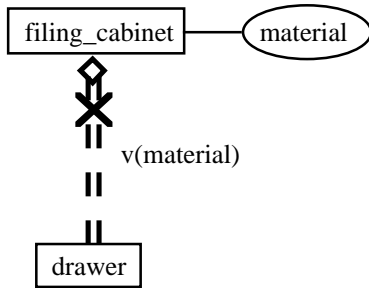


Figure 17: Downward propagation

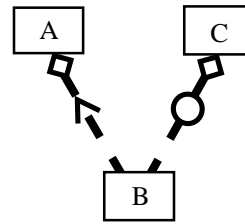


Figure 18: An inconsistent part schema

$a$ ? Since  $b$  is dependent on it,  $b$  must be deleted, too. However, deleting  $b$  would leave  $c$  without an essential part, which is a violation of the essentiality constraint. So,  $b$  must not be deleted. Thus,  $b$  must and must not be deleted—a contradiction. ■

## 5 Conclusion

In this paper, we have presented a comprehensive part model for OODB systems. This model incorporates a variety of part relationships with characteristics such as exclusiveness/sharing, single-/multi-valuedness, cardinality range-restriction, ordering of definite or indefinite number, essentiality, dependency, and value propagation. We have distinguished two kinds of exclusiveness, global exclusiveness and class exclusiveness. Dependency in our model can be from the part to the whole, or vice versa. General definitions for upward and downward value propagation along the part relationship were given. A basis for the realization of these relationships within an existing OODB was discussed.

To complement the formal definitions, we have presented graphical symbols for each of the part relationships. These symbols expand a graphical schema representation for OODBs that we have previously devel-

oped [12]. Each was designed to be intuitive, making part schemata easier to read and understand.

In a future paper, we will be presenting a complete discussion of how the different part relationships can be combined to form others. We will also be investigating the way the relationships can be used together within the overall database schema. As was demonstrated, there do exist inconsistent conceptual part models. Query support for our model is also a topic under investigation. Of particular interest is how transitivity [5, 22] plays a role in queries against the part hierarchy. A third kind of exclusiveness, which we referred to as *intra-object* exclusiveness, is another issue that we are currently pursuing.

## References

- [1] J. Banerjee et al. Data model issues for object-oriented applications. In M. Stonebraker, editor, *Readings in Database Systems*, pages 445–456. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
- [2] R. Bretl et al. The GemStone data management system. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Appli-*

- cations, pages 283–308. ACM Press, New York, NY, 1989.
- [3] P. P.-S. Chen. The Entity-Relationship Model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [4] P. Coad and E. Yourdon. *Object-Oriented Analysis*. Yourdon Press Computing Series. Prentice Hall, Englewood Cliffs, NJ, second edition, 1991.
- [5] D. A. Cruse. On the transitivity of the part-whole relation. *Journal of Linguistics*, 15(1):29–38, 1979.
- [6] D. H. Fishman et al. Overview of the Iris DBMS. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 219–250. ACM Press, New York, NY, 1989.
- [7] J. Geller. *A Knowledge Representation Theory for Natural Language Graphics*. PhD thesis, SUNY Buffalo CS Department, 1988. Tech. Report 88-15.
- [8] J. Geller. A graphics-based analysis of part-whole relations. Research Report CIS-91-27, CIS Department, NJIT, Nov. 1991.
- [9] J. Geller. Propositional representation for graphical knowledge. *Int. J. Man-Machine Studies*, 34(1):97–131, 1991.
- [10] J. Geller and S. Shapiro. Graphical deep knowledge for intelligent machine drafting. In *Tenth Int'l Joint Conference on Artificial Intelligence*, San Mateo, CA, 1987. Morgan Kaufmann Publishers, Inc.
- [11] M. Halper, J. Geller, and Y. Perl. “Part” relations for object-oriented databases. In *Proc. of 11th Int'l Conference on the Entity Relationship Approach*, Karlsruhe, Germany, Oct. 1992.
- [12] M. Halper, J. Geller, Y. Perl, and E. J. Neuhold. A graphical schema representation for object-oriented databases. In *IDS92, Int'l Workshop on Interfaces to Database Systems*, Glasgow, Scotland, July 1992. To be published by Springer Verlag.
- [13] W. Horak and G. Krönert. An object-oriented office document architecture model for processing and interchange of documents. In *Second ACM-SIGOA Conference on Office Information Systems*, pages 152–160, Toronto, Canada, June 1984.
- [14] IEEE Computer Society. *Proc. of Second Int'l Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Gaithersburg, MD, Oct. 1989.
- [15] W. Kim, E. Bertino, and J. F. Garza. Composite objects revisited. In *Proceedings of the 1989 ACM SIGMOD Int'l Conference on the Management of Data*, pages 337–347, Portland, OR, June 1989.
- [16] B. K. MacKellar and J. Peckham. Representing design objects in SORAC: A data model with semantic objects, relationships and constraints. In *Second International Conference on Artificial Intelligence in Design*, Pittsburgh, PA, June 1992.
- [17] G. T. Nguyen and D. Rieu. Representing design objects. In J. Gero, editor, *AI in Design '91*. Butterworth-Heinemann Ltd., 1991.
- [18] J. Peckham and F. Maryanski. Semantic data models. *ACM Comp. Surveys*, 20(3):153–189, Sept. 1988.
- [19] E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, Inc., New York, NY, second edition, 1991.
- [20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenzen. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ, 1991.
- [21] J. Smith and D. C. P. Smith. Database abstractions: Aggregation and generalization. *ACM TODS*, 2(2):105–133, 1977.
- [22] M. E. Winston, R. Chaffin, and D. Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11(4):417–444, 1987.
- [23] D. Woelk, W. Kim, and W. Luther. An object-oriented approach to multimedia databases. In *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 311–325, Washington, D.C., May 1986.
- [24] S. B. Zdonik and D. Maier. Fundamentals of object-oriented databases. In S. B. Zdonik and D. Maier, editors, *Readings in Object-Oriented Database Systems*, pages 1–32. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.