An Object-Oriented Conceptual Model for the Representation of Geographic Information

Eliseo Clementini, Paolino Di Felice, and Alessandro D'Attri

Dipartimento di Ingegneria Elettrica,
Università dell'Aquila, I-67100 L'Aquila, ITALY

Abstract

The interest of the community towards geographic databases is large. The drawbacks of traditional database systems when used for handling not structured data have been largely pointed out in the literature. At present, big research efforts are concentrated on the investigation of the usability of the object-oriented approach for developing a new generation of powerful geographic database systems.

The complexity of geographic systems, in terms of relationships to be modelled and operations to be performed on data, is well-known; therefore a conceptual analysis of the specific situation is a good strategy for handling the complexity of the implementation. In this paper, we propose an object-oriented conceptual model tailored for organizing and representing basic map elements and their relationships, as well as operations of interest for the management of geographic data.

Our aim is to provide designers with a conceptual tool useful for organizing their knowledge about a geographic application in terms of basic concepts of the object-oriented paradigm, namely classes, instances, and methods. We hope such a tool will facilitate the dialogue between designers and developers.

1. Introduction

Spatial data models are growing in popularity (e.g., [1]). So far, various proposals have appeared in the literature, some of them with special reference to the management of spatial information contained in geographic maps (e.g., [2-5]). As reported, for instance, in [6], most proposed spatial data models are obtained by extending the relational one with additional features for managing pictures. Although the relational data model is simple and easy to implement, it is not appropriate for handling spatial data.

At present, there is an increasing interest towards the object-oriented approach (e.g., [7-9]) as a tool for overcoming the drawbacks of traditional data models. Main merits of object-oriented models over traditional ones concern the modelling power, because objects reflect a "natural" view of the world we are modelling in our software, the reusability and extensibility of software components and applications, due to the data abstraction, inheritance, and polymorphism capabilities, and the possibility of merging most of the application into the database schema, due to the encapsulation of data and procedures. Of course, the OO technology has been an expanding area of research only in the last decade and many problems have still to be solved [10-11].

In the field of traditional databases, the adoption of object-oriented data models and programming languages is not new (e.g., [12-14]). On the opposite, investigating the usability of the object-oriented approach to manage spatial data is a relatively new field. In fact, only recently, Mohan and Kashyap [6] proposed an object-oriented formalism for the representation of spatial knowledge. They coupled their formalism with facilities for the deductive inferring via the use of predicate logic and pattern matching paradigm.

In this paper, we use the object-oriented approach as a uniform conceptual frame for modelling most of the knowledge associated to maps, as well as operations on spatial data. In particular, we concentrate our attention on the management of geographic data; nevertheless, most of our considerations are reusable when dealing with other kinds of spatial data.

Typically, geographic applications are large and complex, therefore a conceptual analysis of the specific situation is a good strategy for handling the complexity of the implementation. The motivation of the introduction of semantic data models [15] and OO data models [16-18] is the design of a higher level database model that will enable the database designer to naturally and directly incorporate more of the semantics of a database into its schema. Some authors (e.g., [19]) define with the term "object-oriented" a category of models which encompasses both semantic data models (which are viewed as structurally OO) and OO data models (which are viewed as behaviorally OO). In fact, the main difference which characterizes OO data models with respect to semantic data models is the addition of data abstraction and operations (i.e., methods). These two features make the OO model an extensible model. There are three major reasons why we introduce an object-oriented conceptual data model specifically tailored for organizing and representing knowledge about maps, namely:

• homogeneity: the final aim of our research activity is to use the object-oriented paradigm for conceptualizing, implementing, browsing, and querying a geographic database;

• reusability: we provide the designer with a tool useful for organizing his/her knowledge about the application in
The remainder of this paper is organized as follows. In Sections 2, 3, and 4, we propose an object-oriented conceptual model tailored for organizing and representing basic map elements and their relationships, as well as operations of interest for the management of geographic data.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

2. Geographic elements and their relationships

In the conceptual data model given in this section, geographic elements are represented by objects. There are two kinds of objects: class objects (C) and instance objects (I). Examples of geographic class objects are Continent, Country, River, Lake, while examples of instance objects are America, USA, Mississippi, Ontario.

Among objects there exist binary relationships. The formalism <01, r, 02> denotes that the object 01 has the relationship r with the object 02; we call this triplet a fact. Facts of the kind <C1, r, C2> provide an intensional description of data, while facts of the kind <I1, r, I2> provide an extensional description of data (e.g., [21]). Notice that facts of the kind <I1, r, C> (or <C, r, I>) act as a link between these two data description levels. Four different kinds of facts will be analyzed with respect to their relevance in the geographic context: classification, generalization, aggregation (conceptual abstraction primitives), and location (spatial abstraction primitives).

For example, <C1, is_an_instance_of, C2> is a classification fact [22]. In particular <l, is_an_instance_of, C> associates instance objects with their class. As an example, we have: <Mississippi, is_an_instance_of, River>. On the other hand, <C1, is_an_instance_of, C2> means that C2 is a metaclass of C1.

There are situations where an object may be an instance of many classes. For example, the Hawaii are at the same time an archipelago and an American state. Therefore, it turns out: <Hawaii, is_an_instance_of, Archipelago> and <Hawaii, is_an_instance_of, State>. Classification facts define a hierarchy on classes and instances (Fig.1). In the figure, ovals denote objects, while oriented lines denote relationships.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.

The conceptual model described in this paper is an improved version of that given in [20]. Specifically, in the present paper the initial data model is enhanced with a set of properties which specify the data structure of the model and allow to derive new "facts" starting from known "facts"; moreover, operations on geographic data are formally introduced. The role of operations in our model will be pointed out in Section 4.

Object-oriented geographic databases are made up of large persistent objects. A central issue in the design of such systems is what knowledge should be represented in the database (inside the structure of objects) and what should be deduced dynamically (via methods). Starting from the conceptual tools discussed in previous sections, Section 5 goes towards an object-oriented implementation of geographic databases. Explicit representation versus computed knowledge is analyzed in order to give general hints to the database designer about the way of using our model.

Conclusions are given in Section 6.
fact <Rome, is_an_instance_of, Capital>, and <55.155.993 inhabitants, is_a_part_of, Italy> where the role of "55.155.993 inhabitants" is specified by the fact <55.155.993 inhabitants, is_an_instance_of, Population>.

Aggregation facts define hierarchies on classes and instances (Fig.3). The is_an_instance_of links between the extensional and intensional part of Fig.3 are omitted, since the role of the instances is quite clear. Notice that in Fig.3, each intensional is_a_part_of link has a corresponding extensional link. In more general terms, given the facts <l, is_an_instance_of, C> and <C1, is_a_part_of, C>, it must exist one (and exactly one) instance I1 having the role <I1, is_an_instance_of, C1> such that <I1, is_a_part_of, I> holds.

Fig.3. An example of aggregation hierarchy at the intensional (a) and extensional (b) level.

In many situations, a class may be a component of more than one class (Fig.4(a)). In general, this is not true at the instance level (Fig.4(b)).

The three kinds of facts described so far are not peculiar features of geographic applications (in fact, they occur each time we model a fragment of reality). For such a reason, we did not discuss them in a detailed way and we did not introduce the dual facts, namely instantiation, specialization, and disaggregation (e.g., [21-23]).

Facts modelling major spatial properties of geographic elements are, instead, essential when dealing with geographic applications. Specifically, it is essential to be able to represent in our model geographic situations such as the following: a state belongs to a country (continent); a lake belongs either completely or partially to a state (country, continent); a river crosses either one or more states (countries, continents); a city belongs to a state (country, continent); a highway crosses a state (country); and so on.

Below, we show that conceptual abstraction primitives are not appropriate for modelling the geographic situations mentioned above. To substantiate this claim, instead of taking into account directly geographic elements (e.g., continent, country, state, lake, river, city, and so on), it is convenient to refer to their geometric counterpart (i.e., either a point, a line, or an area). In fact, in this way the claim can be proved simply by showing that classification, generalization, and aggregation are not adequate for modelling the containment (either complete or partial) of a geometric element into an area.

As a first case, let us refer to the containment of a line in an area. Many geographic situations, such as a country belonging (either entirely or partially) to a continent, are geometrically modelled by this case. In order to prove that it is not possible to represent such a spatial property by means of the three relationships discussed so far let us suppose that the classes Continent and Country have been defined. We need a link between a country and a continent both at the intensional and extensional level.

1. The is_an_instance_of link only establishes a relationship between a class and its instances. Hence, the is_an_instance_of link is not suitable for our purposes.

2. Let us suppose, by contradiction, that <Country, is_a_subclass_of, Continent>. This fact states that Continent is a more general concept than Country and, hence, it states that Country is a special kind of Continent. This is clearly false. Furthermore, the is_a_subclass_of link does not have any meaning at the instance level.

3. Let us consider, by contradiction, the fact <Country, is_a_part_of, Continent>. At the extensional level, we would have many instances of the class Country for each instance of the class Continent, such as <USA, is_a_part_of, America>, <Canada, is_a_part_of, America>, etc. This does not match the way the is_a_part_of link has been defined previously.

As a second case, let us refer to the containment of a line in an area. A geographic situation modelled in this way is a river crossing a state (in particular, we have that a river belongs
either entirely or partially to a state). This is another example of spatial property that is not possible to represent by means of the three relationships discussed previously. This claim may be substantiated proceeding as in the first example.

As a third situation, let us refer to the case of a point contained in an area. Geographic examples (correctly stated only when the map scale is large enough) are a city and the summit of a mountain contained in a state. There are cases where a point is shared by different geographic areas such as, for instance, in the case of the Mont Blanc belonging partly to Italy and partly to France. Also here, we need a relationship different from the three already given.

We introduce the fact \(<01, \text{is-in}, 02>\) (where 01 and 02 are either both classes or both instances) to represent in a uniform way spatial properties above. We name \(<01, \text{is-in}, 02>\) a location fact. At the intensional level, we can model some examples above as follows: \(<\text{Country}, \text{is-in}, \text{Continent}>\), \(<\text{River}, \text{is-in}, \text{State}>\), and \(<\text{Mountain}, \text{is-in}, \text{Country}>\), whereas at the extensional level we can model them as follows: \(<\text{Canada}, \text{is-in}, \text{America}>\), \(<\text{Mississippi}, \text{is-in}, \text{Louisiana}>\), \(<\text{Mississippi}, \text{is-in}, \text{Arkansas}>\), and so on. Location facts define a hierarchy on classes and instances (Fig.5). In general, a single link at the intensional level (Fig.5(a)) originates many links at the extensional level (Fig.5(b)).

Fig.5. An example of a location hierarchy at the intensional (a) and extensional (b) level.

The orientation of the is-in link is fixed in accordance with the meaning commonly associated to geographic situations. For example, if the territory of a state and the surface of a lake overlap each other, we say that the lake belongs (at least partly) to the state (and we write \(<\text{Lake}, \text{is-in}, \text{State}>\) ), but we do not say that the state belongs to the lake, as this would be a nonsense. Situations of "partial" containment may cause one-to-many is-in relationships at the instance level (Fig.6-7).

Fig.6 points out a situation where one-to-many is-in links are more usual; in fact, a river usually crosses many states. Fig.7 points out a situation where one-to-many is-in links may be considered an exception; in fact, the USSR are one of the few countries in the world, whose territory geographically belongs to two continents, i.e., Europe and Asia. One-to-many relationships may also exist at the class level; Fig.8(a) shows one of such examples.

In this paper, we use the is-in relationship in a more general sense than the "belongs_to" relationship proposed in [6], where it is used only to model the strict containment of a geographic area in another one.

It is worthwhile to notice that location facts allow to represent spatial data at various levels of spatial abstraction. For instance, classes State, Country, and Continent correspond to spatial data defined at different levels of spatial resolution and related each other by the following facts: \(<\text{State}, \text{is-in}, \text{Country}>\) and \(<\text{Country}, \text{is-in}, \text{Continent}>\). The dual relationship of is-in is the contains relationship. We name \(<01, \text{contains}, 02>\) (where 01 and 02 are either both classes or both instances) a contains fact. By using the contains relationship, at the intensional level we can model facts like: \(<\text{Continent}, \text{contains}, \text{Country}>\) and \(<\text{State}, \text{contains}, \text{River}>\), while at the extensional level we can model facts like: \(<\text{America}, \text{contains}, \text{USA}>\), \(<\text{America}, \text{contains}, \text{Canada}>\), ..., and \(<\text{Arkansas}, \text{contains}, \text{Mississippi}>\), \(<\text{Arkansas}, \text{contains}, \text{White River}>\), ...
3. Properties of facts

In this section, we list a set of formal properties that the abstraction primitives of previous section must satisfy. The aim of these properties is to specify the data structures of the model as well as to derive new facts starting from known facts. Specifically, we show that the model allows to organize classification, generalization, aggregation, and location facts (both at the intensional and extensional level) as directed acyclic graphs. Furthermore, "transitivity" and "inheritance" properties provide a deductive mechanism equivalent to that proposed in [6].

Below, we use $\mathcal{I}$ to denote the set of all instances and $\mathcal{C}$ to denote the set of all classes, with $\mathcal{I} \cap \mathcal{C} = \emptyset$.

Properties of classification facts:

1. **required classification for instances:**
   for every $I \in \mathcal{I}$, there exists $C \in \mathcal{C}$ such that $I$ is-an-instance-of, $C$;

2. **irreflexivity:**
   for every $C_1, C_2 \in \mathcal{C}$, if $C_1$ is-an-instance-of, $C_2$, then $C_1 \neq C_2$;

3. **acyclicity of the classification hierarchy:**
   for every $C \in \mathcal{C}$, it does not exist a sequence of classes $C_1, C_2, ..., C_N \in \mathcal{C}$, where $N > 0$, such that $C_i$ is-an-instance-of, $C_j$, $i < j$.

Property 1 states that every instance is a member of at least one class. This assumption corresponds to refer to a strictly typed data model as meant, for instance, in [21, p.8]. Properties 2 and 3 guarantee the absence of cycles in the classification hierarchy.

Properties of generalization facts:

4. **root of the generalization hierarchy:**
   there exists the class Object $\in \mathcal{C}$ such that, for every $C \in (C - Object)$, $C$ is-a-subclass-of, Object;

5. **irreflexivity:**
   for every $C_1, C_2 \in \mathcal{C}$, if $C_1$ is-a-subclass-of, $C_2$, then $C_1 \neq C_2$;

6. **transitivity:**
   for every $C_1, C_2, C_3 \in \mathcal{C}$, if $C_1$ is-a-subclass-of, $C_2$ and $C_2$ is-a-subclass-of, $C_3$, then $C_1$ is-a-subclass-of, $C_3$.

Together, Properties 5 and 6 guarantee the acyclicity of the generalization hierarchy. Furthermore, this hierarchy has the class Object as a single root (Property 4).

Properties of aggregation facts:

7. **irreflexivity:**
   (a) intensional level:
   for every $C_1, C_2 \in \mathcal{C}$, if $C_1$ is-a-part-of, $C_2$, then $C_1 \neq C_2$;

(b) extensional level:
   for every $I_1, I_2 \in \mathcal{I}$, if $I_1$ is-a-part-of, $I_2$ then $I_1 \neq I_2$.

8. **acyclicity of the aggregation hierarchy:**
   for every $C \in \mathcal{C}$, it does not exist a sequence of classes $C_1, C_2, ..., C_N \in \mathcal{C}$, where $N > 0$, such that $C_i$ is-a-part-of, $C_j$, $i < j$.

9. **uniqueness of aggregation at the extensional level:**
   for every $C, C_1 \in \mathcal{C}$ and for every $I \in \mathcal{I}$, if $I$ is-an-instance-of, $C$ and $C_1$ is-a-part-of, $C$, then there exists and it is unique $I_1 \in \mathcal{I}$, where $I_1$ is-an-instance-of, $C_1$, such that $I_1$ is-a-part-of, $I$.

Together, Properties 7 and 8 guarantee the acyclicity of aggregation hierarchies (both at the intensional and extensional level). It follows that "recursive aggregation facts" (see, for instance, [21]) can not be represented in our model. However, this kind of facts are not relevant in the geographic context.

Property 9 establishes the correspondence between a class and its instances in the number of component objects. However, Property 9 keeps open the possibility that an instance may have other components that have no counterpart at the class level (Fig.9). This allows to localize at the instance level specific features that may be not convenient to be modelled at the class level and, hence, to be imposed as a common features of all instances.

![Fig.9. A visualization of Property 9.](image)

Properties of location facts:

10. **irreflexivity:**
    for every $C_1, C_2 \in \mathcal{C}$, if $I_1$ is-in, $C_2$, then $I_1 \neq C_2$;

11. **transitivity at the intensional level:**
    for every $C_1, C_2, C_3 \in \mathcal{C}$, if $I_1$ is-in, $C_2$ and $C_2$ is-in, $C_3$, then $I_1$ is-in, $C_3$;
12. partial transitivity at the extensional level:

for every $C_1, C_2, C_3 \in \mathcal{C}$ and for every $I_1, I_2 \in \mathcal{I}$, if $<I_1, \text{is-in}, C_2>$, $<C_2, \text{is-in}, C_3>$, $<I_1, \text{is-an-instance-of}, C_1>$, $<I_2, \text{is-an-instance-of}, C_2>$, and $<I_1, \text{is-in}, I_2>$, then there exists at least one $I_3 \in \mathcal{I}$, where $<I_2, \text{is-in}, I_3>$ and $<I_3, \text{is-an-instance-of}, C_3>$, such that $<I_1, \text{is-in}, I_3>$.

13. required location:

(a) intensional level:

for every $I_1, I_2 \in \mathcal{I}$, where $<I_1, \text{is-in}, I_2>$, there exist $C_1, C_2 \in \mathcal{C}$, where $<I_1, \text{is-an-instance-of}, C_1>$ and $<I_2, \text{is-an-instance-of}, C_2>$, such that $<C_1, \text{is-in}, C_2>$.

(b) extensional level:

for every $C_1, C_2 \in \mathcal{C}$ and for every $I_1 \in \mathcal{I}$, where $<I_1, \text{is-in}, C_2>$ and $<C_1, \text{is-an-instance-of}, C_1>$, there exists at least one $I_2 \in \mathcal{I}$, where $<I_2, \text{is-an-instance-of}, C_2>$, such that $<I_1, \text{is-in}, I_2>$.

Together, properties 10 and 11 guarantee the acyclicity of the location hierarchy at the intensional level. These properties, together with property 13(a), assure the acyclicity also at the extensional level. Property 13(a) states that if a location fact exists at the instance level, then it must exists also at the class level. This property may be regarded as a "strict typing" of location facts at the extensional level. Property 13(b) is the dual of Property 13(a). At the extensional level, the transitivity of location facts is not valid in general. Property 12 establishes the existence of "paths" in the location hierarchy where transitivity is valid. This may be clarified by means of an example (Fig.10): from $<\text{Siberia, is-in, USSR}>$, $<\text{USSR, is-in, Asia}>$, and $<\text{USSR, is-in, Europe}>$, we may deduce only $<\text{Siberia, is-in, Asia}>$, since the fact $<\text{Siberia, is-in, Europe}>$ would be clearly false.

Fig.10. An example of partial transitivity of location facts.

Acyclicity of the four kind of facts assures that each hierarchy can be represented as a directed acyclic graph. Below, we give other properties concerning the inheritance of classification, aggregation, and location over generalization:

14. inheritance of classification:

for every $I \in \mathcal{I}$ and for every $C, C_1 \in \mathcal{C}$, if $<I, \text{is-an-instance-of}, C>$ and $<C, \text{is-a-subclass-of}, C_1>$, then $<I, \text{is-an-instance-of}, C_1>$.

15. inheritance of aggregation:

for every $C, C_1, C_2 \in \mathcal{C}$, if $<C_1, \text{is-a-part-of}, C>$ and $<C_2, \text{is-a-subclass-of}, C>$, then $<C_1, \text{is-a-part-of}, C_2>$.

16. inheritance of location:

for every $C, C_1, C_2 \in \mathcal{C}$, if $<C_1, \text{is-in C}_2>$ and $<C, \text{is-a-subclass-of}, C_1>$, then $<C, \text{is-in}, C_2>$.

These kinds of inheritance are illustrated in Fig.11. In such a figure, solid lines denote existing facts, while the dashed line denotes a deduced fact.

Fig.11. Inheritance of classification, aggregation, and location over generalization.

Starting from true facts, transitivity and inheritance allow to deduce other facts still true. We distinguish between weak and strong facts. Weak facts are those that can be deduced by
4. Operations and their properties

In geographic maps, objects are represented in a context extremely rich of relationships among them. The abstraction primitives of our model allow to represent most of them, but not all. Specifically, besides location there are other spatial relationships (e.g., adjacency) not directly supported by our model. This reflects our position: we suggest to manage the geographic situations not covered by location via methods. It follows that in our model operations have a twofold role: in fact, they are, at the same time, a tool for inferring spatial knowledge not directly represented in the database and a tool for performing any other kind of computation (e.g., (24-25)).

In this section, our conceptual model is completed with a formalism suitable for specifying operations (called methods) in the model) applicable on objects.

We distinguish between operations applicable on instances (I-methods) and operations applicable on classes (C-methods). We denote the association between an object and a method with a couple <O, method>, which we call an operation set. In particular, the fact that an instance I has an I-method is expressed as: <I, I-method>. Similarly, the fact that a C-method may be applied to a class is expressed as: <C, C-method>. If an I-method is shared among all the instances of a class C, we write: <C, I-method>. Analogously, if a C-method is shared among all classes of a metaclass M, we write <M, C-method>.

With regard to I-methods, an important category of operations that can take place over geographic elements are geometric operations. In general, this kind of operations involves computation on spatial data, which can be either simple or complex, that is, it can be made up of several elementary steps. It is a hard task to point out a complete set of geometric operations, as they strongly depend on the particular objects of the application. Some simple geometric operations are those concerning the computation of the:

- position of a point;
- distance between geometric elements;
- length of a line;
- surface, perimeter, border, and center of an area;
- intersection between two geometric elements;
- union between two geometric elements.

These examples are modelled by I-methods, since they are applicable on instance objects. On the other hand, C-methods represent operations relative to a class object. An incomplete list of operations modelled by C-methods is the following:

- creating a new instance of a class;
- deleting an instance;
- counting all instances of a class;
- adding a fact;
- removing a fact.

Similarly to what already done for facts, below we give a set of properties for methods:

1. required operation set for instances:
   for every CeC and for every IeI, if <C, I-method> and <I, is_an_instance_of, C> then <I, I-method>;
2. required operation set for classes:
   for every C, MeC, if <M, C-method> and <C, is_an_instance_of, M> then <C, C-method>;
3. inheritance over generalization:
   for every C, C1eC, if <C, is_a_subclass_of, C1> and <C1, method> then <C, method>.

5. Explicit representation vs computed knowledge: a discussion

In Section 2, we introduced four kinds of links as a conceptual tool suitable for representing knowledge about geographic maps in terms of (what we name) facts. Hence, in Section 3, a list of properties of facts were given. Specifically, transitivity and inheritance properties constitute a mechanism for deducing new facts starting from known facts. Aim of this section is to discuss the basic issues that should guide a geographic database designer in the decision of what knowledge should be represented in the structure of the objects and what should be computed via methods. Geographic applications are usually very large; moreover, most of them grow in size rapidly. It follows that it is not practical, and sometimes even not feasible, to represent explicitly all the relationships existing in these kinds of applications. A crucial task of the database designer is to fix an appropriate trade-off between these two aspects.

We split the discussion into two parts, the first concerning the facts and the second concerning the links. With respect to strong facts, we suggest representing all of them with the exception of strong location facts, for which the decision should be taken on a case-by-case basis. In fact, location (containment) facts could always be deduced by running programs capable to extract this information from a map. In most cases, verifying the is_in relationship corresponds to perform an intersection operation between two geometric areas. Unfortunately, these computations are time consuming. The opposite strategy would concern in storing explicitly strong location (containment) facts, but this could result in an excessive storage overhead. A possible trade off is the following: given the fact <C1, is_in, C2>, if for each instance I1 of C1 we know that there exists only one instance I2 of C2 such that <I1, is_in, I2>, then we suggest storing these facts in the database. In the opposite case, location facts can be computed. According to this choice, the fact <Country, is_in, Continent> should be stored in the database, whereas the fact <River, is_in, Country> should be computed. A river may cross several countries, therefore, we can not know a priori the number of the is_in links for the class River, as it depends on the particular instance of River. As depicted in Fig.7, there are situations where a country belongs to more than one continent.
however, we know that these cases are rare, hence may be convenient to treat them as exceptions.

With respect to weak facts, we suggest computing them. For instance, this means that given the facts <Lake, is_a_subclass_of, Geographic Area> and <Geographic Area, is_a_subclass_of, Area>, the fact <Lake, is_a_subclass_of, Area> has not to be represented since it can be deduced via transitivity. Similarly, given, for instance, the facts <Arizona, is_an_instance_of, State> and <State, is_a_subclass_of, Geographic Area>, the fact <Arizona, is_an_instance_of, Geographic Area> has not to be represented since it can be deduced via the inheritance of classification over generalization.

With respect to links, the issue concerns if both directions (i.e., a link and its dual) have to be represented. Our position is the following. We propose to store the is_a_subclass_of and the is_an_instance_of links, but not their duals. Moreover, we propose not to store the is_a_part_of link but its dual, this because by representing the is_a_part_of's dual link all the components of an object are collected together and this is essential to get at first glance a precise idea about the structure of an object. Finally, the is_in and contains links can be either represented or deduced, depending on the particular situation.

6. Conclusions

Basic features of the conceptual data model presented in this paper can be stated in terms of simplicity, data structuring flexibility, and object-orientation.

Simplicity. We describe the knowledge associated to geographic maps in terms of four basic facts, resulting in a simple and homogeneous data structure for each category of facts (a directed acyclic graph).

Data structuring flexibility. For any knowledge representation model to be successful, it is essential that its "structures" are consistent with the knowledge of the given world. The object-oriented conceptual model described in this paper provides the appropriate framework for representing knowledge about geographic maps. Specifically, location facts are suitable for capturing the inherent structure of spatial data. In fact, objects (either classes or instances) can be created at various levels of spatial resolution and the information concerning the links between them can be stored along with the objects themselves as facts of the type <O1, is_in, O2>.

Object-orientation. By using our model, the knowledge about geographic maps can be expressed in terms of basic concepts of the object-oriented paradigm, namely classes, instances, and methods. As a consequence the successive object-oriented implementation is facilitated.

As stated in the Introduction, the model presented in this paper is intended to be a conceptual tool for the designer of object-oriented geographic databases, helping in organizing his knowledge about the application. It is worthwhile to notice that our model may be also useful to the designer of the human computer interaction environment, who takes the charge of constructing, among other things, the browsing environment. [27] proofs this statement. In such a paper, authors proposed a browsing technique adopting the four DAGs of Section 3 as "adjacency structures" among objects.

Acknowledgements. This work was supported by the Italian National Council of Research (CNR) under the project "Sistemi Informatici e Calcolo Parallelo - P5 - MULTIDATA".
References


