CERIAS Tech Report 2004-23

AN AUTHORIZATION MODEL FOR GEOGRAPHICAL MAPS

by A. Belussi, E.Bertino, B.Catania, M.L. Damiani, A.Nucita

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

An Authorization Model for Geographical Maps

A. Belussi[1], E. Bertino[2], B. Catania[3], M. L. Damiani[4], and A. Nucita[4]

[1] Dipartimento di Informatica, University of Verona

E-mail: alberto.belussi@univr.it

[2] Cerias and CS Department, Purdue University, West Lafayette IN, USA

E-mail: bertino@cerias.purdue.edu

[3] Dipartimento di Informatica e Scienze dell'Informazione, University of Genova, Italy

E-mail: catania@disi.unige.it

[4] Dipartimento di Informatica e Comunicazione, University of Milano, Italy

E-mail: {damiani,nucita}@dico.unimi.it

Abstract

Access control is an important component of any database management system. Several access control models have been proposed for conventional databases. However, these models do not seem adequate for geographical databases, due to the peculiarities of geographical data. Previous work on access control models for geographical data mainly concerns raster maps (images). In this paper, we present a discretionary access control model for geographical maps. We assume that each map is composed of a set of features. Each feature is represented in one or more maps by spatial objects, described by means of different spatial properties: geometrical properties, describing the shape, extension and location of the objects composing the map, and topological properties, describing the topological relationships existing among spatial objects. The proposed access control model allows the security administrator to define authorizations against map objects at a very fine granularity level, taking into account the various spatial representations and the object dimension. The model also supports both positive and negative authorizations as well as different propagation rules that make access control very flexible.

1 Introduction

Geographical data have a strategic relevance in several contexts; this is the case for example of homeland security applications but also of marketing analysis tools or environmental risks control procedures. In all these applications, the access to geographical information must be controlled at possibly different levels of granularity. A possible naive approach to support access control is to build ad hoc datasets (maps) for each type of access the administrator wants to grant: this has been the cartographic approach applied for many years in the past. Such an approach is not suitable when the user community is large and dynamic

- which is today often the case in Web-based systems. Moreover, such an approach does not support flexible protection granularities and dynamic changes in the access control policies. The introduction of integrated GIS systems, characterized by high level comprehensive data models, is today making possible the development of advanced access control models which go beyond such naive approach. However, despite the importance of data protection, no efforts have been devoted to the investigation of access control models and systems for geographical data stored in GIS systems.

Several access control models have been proposed for conventional databases. However, these models are not adequate for geographical databases, due to the peculiarities of geographical maps, where geographical objects can be represented with different dimensions and the accesses can also be driven by the reference space (i.e., authorization to access only data concerning geographical entities in a given region). Moreover, geographical data can be represented using different approaches. The users of a GIS system usually recognize geographical data from the existence of a geometry describing the shape, extension and location of some geographical objects (features). However, geographical data can be represented also in other forms, for example by using a set of topological relations (topological representation), that specify the adiacency, the disjointness or other kind of interaction between two features. Those various representations should be taken into account when defining an access control model.

Previous work on access control models for geographical data has mainly dealt with raster maps (images) [1, 6], focusing on data confidentiality when sensible information can be revealed by high resolution image satellites. In such model, each protected object is basically an image or portion of an image. Thus, the model does not support neither the vector-based representation of entities nor the topological one. As such, the model is not adequate for usage in current GIS and spatial DBMS. Moreover, it only deals with read privileges, thus, it is not adequate for dynamic applications that require data modifications. In [3, 4], an extension of the classical discretionary access control model is proposed to protect vector-based spatial data against requests issued through a Web service. In such a work, spatial data consists of objects having a geometry compliant with the OpenGIS simple features model [12]. In the proposed access control model, authorizations on spatial objects should be applied, if necessary, on limited areas (*windows*) smaller than the whole space. As an example, a user may be authorized to insert road objects only if the roads are located in a well defined region. The window defines the geographical scope of the authorization, making authorizations themselves geographical objects which occupy a position in the reference space. Even if the model proposed in [3, 4] identified some interesting requirements for geographical data access control models, the considered object model is quite simple: spatial objects have simple geometries and a unique representation (topological information is not considered). Moreover, the proposed access control model supports only positive authorizations (corresponding to privileges that can be granted). Negative authorizations, corresponding to privileges that must be denied, are not considered.

In this paper, we present an access control model for geographical maps, admitting multiple (vector-based and topological) representations. We assume maps to be represented according to a simplified version of the Layered Spatial Data Model (LSDM) proposed in [2]. Such data model is based on the concept of *feature type*. Examples of feature types are roads, lakes, and so on. Each map contains various instances of the feature types (several roads, several lakes, and so on), each represented with a given

dimension (0 if they are represented as points, 1 if they are lines, 2 if they are represented as regions). Each specific feature is associated inside a map with at least one spatial representation (geometrical or topological). If several representations are provided, they must be consistent. It is important to point out that the considered model supports the representation of topological information independently from the corresponding geometric information. This is an important issue for several applications, for example those concerning transport network management. In these applications, the network graph (thus, the topological representation) is usually known but the exact position of graph nodes (thus, the geometric representation) is not always required.

The proposed access control model takes into account several requirements previously discussed. It allows one to specify authorizations against map objects at a very fine granularity level. It also takes into account the various spatial representations and the object dimensions. For example, under the proposed model, the administrator can authorize a user to see a given object only at 0 dimension and not at higher dimensions, thus hiding detailed information about the object shape. The model also supports both positive and negative authorizations, giving precedence to the negative ones, as well as various propagation rules that make access control very flexible. Moreover, similarly to [3], the model supports the concept of authorization window, specifying the region of space in which the authorization applies.

Since there are some similarities between the model adopted for spatial data and the typical object-oriented data models, in order to develop our access control model we took into account access control models proposed for object-oriented databases [14, 16, 5]. In particular, we borrowed the concepts of weak and strong authorizations from the authorization model of Orion [14]. In the Orion approach, an authorization is strong if it, and any authorization it implies, cannot be overridden by other authorizations. By contrast, authorizations implied by a weak authorization can be overridden.

A limitation of the Orion model, that makes it not suitable for geographical maps, is that authorizations propagate only through objects that are related by the various modeling hierarchies, like the inheritance hierarchy or the composite object hierarchy. Geographical maps, however, require more sophisticated propagation mechanisms able to take into account several object properties, like the spatial layer in which the objects are represented, or the dimensional level of their geometric representations. The proposed model thus provide a propagation mechanism specifically tailored for geographical maps.

The proposed access control model consistently differs from the one presented in [3, 4]. Indeed: (i) the map model is more complex, providing multiple representations (geometrical and topological) of the same map object and multiple representations of the same feature; (ii) in [3, 4] propagation is only allowed along application-dependent privilege hierarchies; on the other hand, in our model propagation rules constitute an invariant part of the access control model and are defined along object and privilege hierarchies; (iii) differently from [3, 4], both positive and negative authorizations, as well as a mechanism to define exceptions to propagated authorizations, are supported.

The paper is organized as follows. In Section 2 we briefly present the adopted map data model. In Section 3 we define the basic elements of the authorization model. Access control model and mechanisms are then presented in Section 4. Finally, Section 6 presents some conclusions and outlines future work.

2 The Topological Spatial Data Model

In this paper we consider a simplified version of the Layered Spatial Data Model (LSDM) presented in [2]. We call this model Topological Spatial Data Model (TSDM), since we keep only the topological layer of LSDM beyond the geometrical one. In TSDM the schema of a spatial database can be defined as a set of feature types and a set of map types. A map type is a set of feature types. A map type contains one or more feature types and a feature type can belong to different map types. Each feature type has some descriptive attributes and one spatial attribute. The spatial attribute of a feature type can be represented in different maps with different dimensions. Formally, a spatial database schema in TSDM can be defined as follows.

Definition 1 (Spatial Database Schema in TSDM) *The schema of a spatial database is a 5-tuple* $S = (\mathcal{E}, n(), Dom_{\mathcal{E}}(), \mathcal{M}, Map())$ *where:*

- $\mathcal{E} = \{E_1, ..., E_k\}$ *is a set of feature type identifiers.*
- $n : \mathcal{E} \rightarrow \mathcal{N}$ *is a function which defines the number of attributes of each feature type* $E_i \in \mathcal{E}$*. The j-th attribute of* E_i *is denoted by* $E_i.a_j$*,* $1 \leq j \leq n(E_i)$*. Each feature type* E_i *has an attribute called identifier denoted as* $E_i.a_0$
- $Dom_{\mathcal{E}} : \mathcal{E} \times N \rightarrow \{D_{number}, D_{string}\}$ *is a partial function which defines the domain of each attribute.*
- $\mathcal{M} = \{M_1, ..., M_h\}$ *is a set of map types.*
- $Map : \mathcal{E} \times \mathcal{M} \rightarrow \{-1, 0, 1, 2\}$ *is a total function which defines if a feature type* E_i *belongs to a map type* M*. In particular, if* $Map(E_i, M_j) = -1$*, the feature type* E_i *does not belong to the map type* M_j*; if* $Map(E_i, M_j) \geq 0$*,* E_i *belongs to* M_j *and the spatial representation of* E_i *in* M_j *is: the set of isolated points of the Euclidean plane* (E^2)*, if* $Map(E_i, M_j) = 0$*; the set of simple polylines of* E^2*, if* $Map(E_i, M_j) = 1$ *and the set of simple polygons of* E^2*, if* $Map(E_i, M_j) = 2$*. Polylines and polygons can be connected or not.*

Feature and map types are also called schema objects. □

The fact that a feature type has one or more geometrical domains associated with it in different map types does not imply that a feature will have in the database a geometrical representation. Indeed, the spatial representation of a feature can exist in one or both the available layers, that we briefly present here (more details can be found in [2]).

- *Geometrical layer* (D_{geo}): in this layer, shape and location on the earth surface of features are represented, in particular geometrical values belong to one of the following sets: the set of points in the Euclidean plane E^2, the set of simple connected or not connected polylines in E^2, and the set of simple polygons of E^2 delimited by a simple closed polyline or by a set of closed polylines (not connected polygons).
- *Topological layer* (D_{topo}): in this layer the spatial properties of each feature are represented simply by describing the topological relations of the feature with other features of the map [8]. The

reference set of topological relations is $\{Disjoint, Touch, In, Contains, Equals, Cross, Overlap\}$.[1] These relations are binary, mutually exclusive (if one is true, the others are false) and they are a refinement of the well-known set of topological relations proposed in [7].

Given a TSDM schema, a TSDM instance can be defined as follows.

Definition 2 (Spatial Database Instance in TSDM) *The instance of a spatial database schema* $S = (\mathcal{E} = \{E_1, ..., E_k\}, n(), Dom_{\mathcal{E}}(), \mathcal{M} = \{M_1, ..., M_h\}, Map())$ *is composed of:*

- *A set of feature type extensions* $I(\mathcal{E}) = \{I(E_1), ..., I(E_k)\}$. *Each feature* $e \in I(E_i)$ *is a tuple belonging to the domain* $\mathcal{N} \times Dom_{\mathcal{E}}(E_i, 1) \times ... \times Dom_{\mathcal{E}}(E_i, n(E_i))$. *We denote with* $D_{ft} = \{e.a_0 | e.a_o \in I(E_1) \cup ... \cup I(E_k)\}$ *the set of all feature identifiers.*
- *A set of map instances (or simply maps)* $I(\mathcal{M}) = \{I(M_1), ..., I(M_h)\}$. *Each map* $I(M_j)$ *is a set of tuples:* $< f, geo, top > \in D_{ft} \times (D_{geo} \cup \perp) \times (D_{top} \cup \perp)$, *where* $\perp$ *denotes a null value.*

Each tuple is called map object. We denote the instance of a schema S *as* $I(S) = (I(\mathcal{E}), I(\mathcal{M}))$. *Map objects and features are also called* instances. *Extensions of feature types (*feature sets*) and extension of map types (*maps*) are also called* group objects *to emphasize the fact that they have an extension, i.e., a set of instances.* □

Example 1 *Consider a geographical database representing the railway network of Lombardy, the administrative areas (province) of Lombardy, the counties of Lombardy, and the accidents on the railway network. The TSDM schema contains four feature types:* $\mathcal{E} = \{Railway, Accident, Region, County\}$. *Suppose that* $\mathcal{M} = \{Lomb\_rail, Lomb\_admin\}$. *The feature types Railway and Accident belong to the map schema Lomb_rail with dimension 1 and dimension 0, respectively. The feature types Region and County belong to the map schema Lomb_admin with dimension 2. Moreover, suppose that all feature types have an attribute Name* (a_1) *representing the feature name, and an attribute N* (a_2) *representing: the number of inhabitants for feature types Region and County, the number of tracks for Railway, and the accident type for Accident.*

An instance of this schema can contain for example: a complete geometrical representation of the Railway features (and therefore, also a topological one, since the topological representation can be inferred from the geometrical one) (Fig. 1) and a topological representation of Region and County features (Fig. 1). When an accident occurs, either a geometrical representation (a point) can be inserted in the database or a topological one, represented by an In relation between the occurred accident and the railway network. For this example, we assume that the geometrical representation, and therefore also the topological one, are provided. □

In [2], a query language for TSDM has also been proposed. It is an algebra with the following types of operators:

[1] Intuitively, given two feature f_1 and f_2, we say that f_1 *Touch* f_2 if they intersect, but their interiors do not; f_1 *In* f_2 if their interiors intersect and f_1 is inside f_2; f_1 *Contains* f_2 if f_2 *In* f_1; f_1 *Overlap* f_2 if their interiors intersect, f_1 *In* f_2 and f_1 *Contains* f_2 do not hold, and f_1, f_2 have the same dimension of their intersection; f_1 *Cross* f_2 if their interiors intersect, f_1 *In* f_2 and f_1 *Contains* f_2 do not hold, and f_1, f_2 and their intersection have different dimensions.

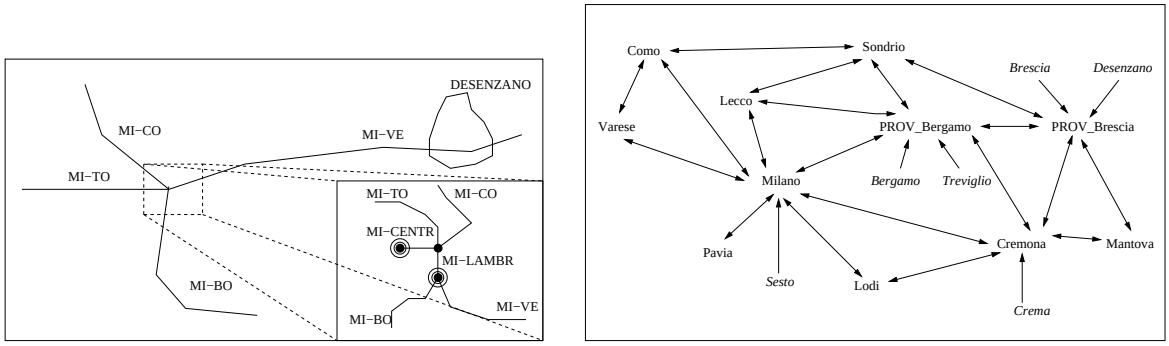

Figure 1: On the left: Example of a geographical database: the railway network of Lombardy. On the right: Topological relations among the features of the *Region* and the *County* feature types (in italics). A double arrow represents a Touch relation, while a single arrow represents a In relation.

- **Feature-based operators**: they are applied to feature sets and produce feature sets; they are very similar to the operators of the relational algebra. Examples of feature based operators are traditional relational operators like *selection* ($\sigma_F(E_i)$), *projection* ($\Pi_{\overline{X}}(E_i)$), *Join* ($E_i \bowtie_F E_j$), and the usual set based operators (union $\cup$, difference $\setminus$, intersection $\cap$).

- **Map-based operators**: they are applied to maps and produce new maps. Among them, we recall: *map selection* ($\sigma_F^M(M_i)$), that takes a map instance $I(M_i)$ and selects all the map elements satisfying a certain formula F; *semi-join* ($M_i \ltimes_F^M M_j$), that takes two map instances $I(M_i)$ and $I(M_j)$ and returns a new map instance containing all the elements of the first map instance related as specified in F to some element of the second map instance; usual set based operators (*union*, *intersection*, *difference*) can be applied on maps.

- **Mixed operators**: they are applied to maps and feature sets and produce either maps or sets of features. Among them, we recall: *mixed projection* ($\Pi_{ft}^x(M_i)$), that takes a map instance $I(M_i)$ and returns the set of features with feature type ft contained in the map; *mixed join* ($M_i \bowtie_F^x M_j$), that takes two map instances $I(M_i)$ and $I(M_j)$ and returns a new set of features obtained by combining together the features associated with pairs of map elements, satisfying condition F.

Example 2 *Consider Example 1. The following are examples of queries, the first, Q_{map}, producing a map, the second, Q_{Fset}, a set of features:*

- Q_{map} : *find all the map objects representing the 'MI-VE' railway from Milan to Venice.*
  $\sigma^M_{Railway.Name='Mi-VE'}(I(Lomb\_rail))$.

- Q_{Fset} : *find all the counties with more that 100,000 inhabitants.*
  $\sigma_{County.N \geq 100{,}000}(I(County))$. □

3 Basic components of the authorization model

The proposed access control model relies on the classical discretionary model centered on the notion of authorization [14]. An authorization in its simplest format consists of a triple specifying: *the subject* of the authorization, i.e. who benefits from it, for example a user or group of users; *the object*, i.e. the logical data structure that needs to be protected, for example a map; and *the privilege*, i.e. the kind of operations that can be performed on the object. The object and the privilege are strictly dependent on the data model. In our model, additional components include the *authorization sign*, specifying whether the privilege has to be granted (+) or denied (-); the *authorization type*, specifying whether the authorization can be overridden (weak) or not (strong); the *window*, i.e., a spatial object representing the region of space over which the authorization can be granted; the *query*, further restricting the set of objects over which the privilege is granted or denied; the *grantor*, i.e., the user assigning the authorization to the subject; the *grant option*, which when true denotes that the subject itself can further grant the (positive) authorization to other users.

In the following, all these components are defined in more detail.

3.1 Subjects and objects

Subjects. Subjects are all users that interact with the system. The access control model we are going to present does not support groups and roles, even if it can be easily extended to deal with them. In the following, we denote with U the set of all the users.

Objects. In a TSDM database, both schema, group, and instance objects have to be protected. Schema objects can be protected with respect to operations concerning access to their definition (metadata). Instance objects can be protected with respect to selection, deletion, and update. Group objects can be protected with respect to operations concerning their extensions (insertion, deletion, update, selection).

It is important to note that, even if, similarly to the relational context, queries are defined against maps and feature sets, unlike the relational context, in a geographical database it is very important to deal with single instances. Indeed, the spatial representation of a feature can be very detailed and thus can be of great interest to a specific end user.

In the following, given a TSDM schema $S = (\mathcal{E}, n(), Dom_{\mathcal{E}}(), \mathcal{M}, Map())$ and its instance $I(S) = (I(\mathcal{E}), I(\mathcal{M}))$, we denote: with O_{FT} and O_{MT} the set of feature types $\mathcal{E}$ of S and the set of map types $\mathcal{M}$ of S respectively, with O_M and O_{FS} the set of maps $I(\mathcal{M})$ of $I(S)$ and the set of feature sets $I(\mathcal{E})$ of $I(S)$ respectively, with O_{MO} and O_F the set of all map objects of the maps $I(\mathcal{M})$ of $I(S)$ and the set of all features of the feature sets $I(\mathcal{E})$ of $I(S)$ respectively.

3.2 Privileges

Privileges can be classified according to the object to which they can be granted (see Table 1). Various privileges can be assigned to instance objects, corresponding to the various operations that can be executed on them: selection, deletion, and update of an instance. We call these privileges *instance privileges*.

Instance privileges can also be assigned to group objects with the following meaning: an instance privilege p assigned to a group object o is propagated, following a precise rule, to the instance objects belonging to o. An additional privilege is assigned to group objects, in order to insert a feature into a feature set or to assign an object to a map (*insertion privileges*). Finally, for schema objects, we consider access to their definition as a privilege (*schema privileges*). In the following, we denote with P the set of all the privileges.

Example 3 *Consider Example 1. The following are some examples of possible privileges:*

- *Privilege* $select_M(1, GEO)$*, granted on Lomb_rail map, allows the user to read the geometrical representation of the Lomb_rail map objects with dimension 1 (i.e., a railway).*
- *Privilege* $assign_M(0)$*, granted on Lomb_rail map, allows the user to insert a map object of dimension 0 (i.e., an accident).*
- *Privilege* $update_F(2, SPACE)$ *on the County feature type allows the user to update the spatial representation of all the County features in each map in which they appear with dimension 2 (i.e, map Lomb_admin).*
- *Privilege* $insert_F$ *on the County feature type allows the user to create a new instance of the County feature type (with no spatial extension).*
- *Privileges* $select\_schema_{FT}$ *and* $select\_schema_{MT}$*, on feature type Railway and map type Lomb_admin, respectively, allow the user to access the meta data concerning the schema of Railway feature type and Lomb_admin map type, respectively.* □

According to Table 1, some privileges allow one to read, delete, or update spatial information ($assign_M$, $delete_F$, the instance privileges of the form $p(d)$, and of the form $p(d,t)$ with $t \in \{SPACE, GEO, TOPO\}$). Those privileges are called *spatial privileges*. All the other privileges are *non-spatial*. Note that, according to Table 1, non-spatial privileges can only be assigned to features or feature types.

Since not all the types of privileges apply to all possible types of objects, it is useful to introduce a function, called *scope* that, taken a privilege, returns the set of objects to which the privilege can be assigned. According to Table 1, the main problem arises with privileges for map objects, since they must be represented at the layer and the dimension required by the privilege.

Definition 3 (Privilege scope) *Let* $p \in P$*. The scope of* p*, denoted by* $s(p)$*, is the set of objects defined as follows (see section 3.1 for the meaning of* O_x *sets):*

$s(select\_schema_{FT}) = O_{FT}$ $\quad$ $s(select\_schema_{MT}) = O_{MT}$

$s(insert_F) = O_{FT}$ $\quad$ $s(assign_M(d)) = O_M$

$s(select_F(d,t)) = O_{FT} \cup O_F$ $\quad$ $s(update_F(d,t)) = O_{FT} \cup O_F$

$s(delete_F) = O_{FT} \cup O_F$

$s(p_M(d)) = O_M \cup \{o | o =< f, geo, top > \in O_{MO}, dim(o) = d\}$

$s(p_M(d,t)) = O_M \cup \{o | o =< f, geo, top > \in O_{MO}, dim(o) = d,$ *if* $t = geo,\ geo \neq \perp,$ *if* $t = top,\ top \neq \perp\}$ □

Privilege	Description
Object: Feature types	
$select\_schema_{FT}$	It provides the access to feature type schema information, i.e., information concerning descriptive attributes of the considered feature type.
Object: Feature sets (Feature type extensions)	
$insert_F$	Ability to insert a new feature, instance of the considered feature type.
Object: Features	
$select_F(d,t)$	$d \in \{0,1,2,\perp\}$, $t \in \{GEO, TOPO, ALPHA\} \cup schema(ft)$. Ability to read information of type t for the considered feature of feature type ft. If $t \in \{GEO, TOPO\}$, selection is provided in all the maps in which the feature appears with the layer of representation t and with dimension d. If $t = ALPHA$, $d = \perp$ (it is not relevant) and selection is provided on all the alphanumeric attributes of the feature. If $t \in schema(ft)$, $d = \perp$ and selection is provided on attribute t of the feature.
$update_F(d,t)$	$d \in \{0,1,2,\perp\}$, $t \in \{SPACE, ALPHA\} \cup schema(ft)$. Ability to update information of type t for the considered feature. If t is $SPACE$, update is provided on the geometrical representation of the feature in all the maps in which it appears with dimension d. If $t = ALPHA$, $d = \perp$ and update is provided on all the alphanumeric attributes of the feature. If $t \in schema(ft)$, $d = \perp$ and update is provided on attribute t of the feature.
$delete_F$	Ability to delete the considered feature. As a side effect, the feature is deleted also from all the maps it has been assigned to.
Object: Map types	
$select\_schema_{MT}$	It provides the access to schema information for the considered map type, i.e., information concerning the dimension of a feature type representation inside the map with the considered map type.
Object: Maps (Map type extensions)	
$assign_M(d)$	$d \in \{0,1,2\}$ Ability to assign a feature to a map (i.e., to insert a map object) with dimension d, inserting spatial information inside at least one layer of the map.
Object: Map objects	
$select_M(d,t)$	$d \in \{0,1,2\}$, $t \in \{GEO, TOPO\}$. Ability to read spatial information at layer t and dimension d for the considered map object, that must have dimension d (otherwise, the privilege is not considered).
$update_M(d)$	$d \in \{0,1,2\}$. Ability to update the spatial information for the considered map object, that must have dimension d (otherwise, the privilege is not considered).
$delete_M(d)$	$d \in \{0,1,2\}$. Ability to delete a certain map object, that must have dimension d (otherwise, the privilege is not considered). Note that after a deletion the feature still exists but it is no more assigned to the considered map object.

Legenda: $schema(ft)$: schema of the feature type the considered feature is an instance of.

Table 1: Privileges

3.3 Authorization sign and type

In the proposed model, both positive and negative authorizations can be specified (*authorization sign*). A positive authorization establishes that a subject is authorized for a given privilege on a given object, whereas a negative authorization establishes that a subject is denied access to a given object under a given privilege. Thus, a subject u may be denied access to object o because either u has no authorization on o or s has a negative authorization on o. We give precedence to negative authorizations, thus, if u has both a positive and a negative authorization on o, u is denied access to o.

The *authorization type* specifies whether an authorization can be overridden or not. More precisely, *weak* authorizations can be overridden by *strong* authorizations. Consider for example instance autho-

rizations for group objects (maps or feature sets). Such authorizations are propagated from the group objects to their instances. A strong authorization guarantees that after the propagation on instances it cannot be overridden. On the other hand, a weak authorization after propagation can be overridden by other authorizations assigned over features or map objects.

Weak and strong authorizations, when combined with authorization sign, are a useful mechanism for modeling exceptions. For example, if user u can update all instances of a given map m but a certain instance o, we can grant a weak positive authorization to u for updating m (thus, all instance of m can be updated by u) and then a negative update authorization on o.

Weak and strong authorizations can also be defined for instance and schema objects, as well as on group objects and insertion privileges. Even if authorizations for such objects and privileges are not propagated, the authorization type can be used to give precedence to positive authorizations with respect to negative ones, as we will see in Section 4.

3.4 Windows and queries

The window indicates the geographical scope (the where) of the authorization; therefore the region to which the authorization applies. We define a window as an object belonging to the TSDM geometrical domain $D_{polygons}$. As such, the window consists of a (collection) of simple polygons with no holes.

It should be noticed that the notion of window is meaningful only for a subset of the authorization objects and precisely for group objects and instance privileges. Suppose, as an example, that a window is applied to the extension of the *Railway* feature type; in such a case, the corresponding authorization states that privileges, concerning spatial information, can be applied only to the railways located in the specified window region even across different maps. Likewise, a window applied to a map indicates that the privilege holds only on a portion of that map. In both cases, the instances considered to be enclosed in the defined window are those overlapping the window itself.

For feature insertion privilege, the window does not make sense, since features do not have a spatial representation. For maps, it specifies which region of space the inserted map instances must intersect.

In order to specify authorizations for subsets of group object extensions, content-based access control is provided by specifying a TSDM query as part of authorizations. The query restricts the set of objects, intersecting the window, to which the authorization applies.

3.5 Authorization grantor and grant option

The grantor is the subject that granted the authorization. Likewise the classical authorization models, we assume that a subject can delegate the administration of the authorization to some other subject. The mechanism used for delegating such functions is that of the *grant option*. The grant option is expressed as a Boolean variable; if it is true, the subject is authorized to grant/revoke the authorization to/from other subjects. The grant option is specified only for positive authorizations; negative authorizations cannot be delegated.

Object	Privilege	Window	Query
schema	any	$\top$	$\bot$
instance	any	$\top$	$\bot$
group	spatial instance privilege	any	any
group	non-spatial instance privilege	$\top$	any
feature set	insertion privilege	$\top$	$\bot$
map	insertion privilege	any	feature type identifier or $\bot$

Table 2: Authorization conditions

4 The Geographical Access Control Model

In our access control model, an authorization is a tuple containing all the components introduced in Section 3. Not all possible tuples, however, represent authorizations. Indeed, tuple components must satisfy some properties depending on the object and the privilege they consider. First of all, the object must belong to the privilege scope. Moreover, authorizations admit the specification of the window and the query components only for group objects (see Table 2). More precisely, the window can be specified only for spatial privileges (either instance or insertion privileges). The query, for instance privileges, is an expression of the TSDM query language returning a subset of the group object extension. For insertion privileges, the query does not seem reasonable. However, when assigning objects to a map, it could be useful to restrict the privilege to objects with a certain feature type. To model this requirement, the query component of such authorizations is extended to represent a specific feature type.

Definition 4 (Authorization) *An authorization is a tuple of the form* $\langle u, p, pt, g, go, o, t, w, q\rangle$*, where:*

- $u \in U$, $p \in P$, $pt \in \{+, -\}$, $g \in U$, $go \in \{true, false\}$, $o \in O$, $t \in \{st, wk\}$, $w \in D_{polygons} \cup \{\top\}$, $o \in s(p)$, *q is either a query expressed in the TSDM query language having o as a parameter or $q = \bot$ or q is a feature type; $w = \top$ represents the overall space whereas $q = \bot$ represents the identity query, i.e. the query that takes an object and returns the object itself.*
- *if o is an instance or schema object, $w = \top$ and $q = \bot$;*
- *if o is a group object, p is a spatial instance privilege, $w \in D_{polygons} \cup \{\top\}$ and $q(o) \subseteq o$;*
- *if o is a group object, p is a non-spatial instance privilege, $w = \top$ and $q(o) \subseteq o$;*
- *if o is a feature set and p is an insertion privilege, $w = \top$ and $q = \bot$;*
- *if o is a map and p is an insertion privilege, $q = \bot$ or q is a feature type identifier.*

Given an authorization a, we use the dot notation to identify authorization components. □

Example 4 *Let W be a set of authorization windows (for the sake of readability, windows are identified by names): $W = \{Milan\_MetropolitanArea, Milan\_City, Sesto\_County\}$. Consider the authorizations sets presented in Figure 2.*

The authorization set A authorizes BOB (grantor=ADMIN) to perform the following operations: (a_1): read the schema of the feature type Railway; (a_2): read alphanumeric information concerning Railway features; (a_3): read the geometry of Lomb_rail map objects intersecting $Milan\_MetropolitanArea$

SET A

$a_1 =< BOB, select\_schema_{FT}, +, ADMIN, true, Railway, st, \top, \bot >$

$a_2 =< BOB, select_F(\bot, ALPHA), +, ADMIN, false, Railway, st, \top, \bot >$

$a_3 =< BOB, select_M(2, GEO), +, ADMIN, true, Lomb\_rail, st, Milan\_MetropolitanArea, \bot >$

$a_4 =< BOB, update_F(0, SPACE), +, ADMIN, true, Accident, wk, \top, \sigma_{N='wrong\ manouevre'}(Accident) >$

$a_5 =< BOB, update_M(0, SPACE), -, ADMIN, false, Lomb\_rail, st, Sesto\_County, \bot >$

$a_6 =< BOB, insert_F, +, ADMIN, true, Accident, st, \top, \bot >$

$a_7 =< BOB, assign_M(0), +, ADMIN, true, Lomb\_rail, st, \top, Accident >$

SET B

$a_8 =< TED, select_M(2, GEO), +, BOB, false, Lomb\_rail, st, Milan\_City, \bot >$

$a_9 =< TED, update_F(0, SPACE), +, BOB, false, Accident, wk, Milan\_City,$

$\sigma_{N='wrong\ manouevre'\ and\ Name='X'}(Accident) >$

SET C

$a_{10} =< TED, select_M(2, GEO), +, BOB, false, Lomb\_rail, st, \top, \bot >$

Figure 2: Some authorization sets

(window restriction); (a_4, a_5): update the spatial representation of Accident features due to "wrong manouevre" in any map (a_4) except those intersecting the window Sesto_County (a_5) (note that this behavior is possible since a_4 is a positive authorization and a_5 is a negative one); (a_6): insert new Accident features; (a_7): insert map objects representing the spatial location of Accident features in the Lomb_rail map.

The authorization set B authorizes TED (grantor=BOB) to perform the following operations: (a_8): read the geometry of Lomb_rail map objects intersecting Milan_City (window restriction); (a_9): update the spatial representation of the Accident features due to "wrong manouevre" and of name "X" only inside the window Milan_City (window restriction).

The authorization set C authorizes TED (grantor=BOB) to read the geometry of Lomb_rail map objects with no window restriction (a_{10}). □

Authorization $\langle u, p, pt, g, go, o, t, w, q\rangle$ states that g has granted u (denied if $pt = -$) privilege p on a set of objects, depending on o, p, w, and q. We call these objects *authorization extension*. Due to the complexity of the map model, the definition of the authorization extension simplifies the definition of access control mechanisms.

Definition 5 (Authorization extension) *Let $a = \langle u, p, pt, g, go, o, t, w, q\rangle$. The authorization extension of a, denoted by $ext(a)$, is the set of objects defined as follows:*

- *if o is an instance or a schema object, $ext(a) = \{o\}$;*
- *if o is a feature set and p is a non-spatial privilege, $ext(a) = \{o' | o' \in q(o), o \in s(p)\}$;*
- *if o is a feature set and p is a spatial privilege, $ext(a) = \{mo | o' \in q(o),\ \exists m \in O_M, mo =< o'.a_o, geo, top >\in m, mo \in s(map(p)),\ mo\ Intersect\ w\}$, where $map(p)$ converts p in the correspond-*

ing privilege over map objects:[2] $map(select_F(d,t)) = select_M(d,t)$, *where* $t \in \{GEO, TOPO\}$; $map(update_F(d, SPACE)) = update_M(d)$ *and* $map(delete_F) = delete_M(2)$.

The extension in this case coincides with the subset of map objects corresponding to a spatial representation of the correct type (i.e., in the scope of $map(p)$*) of at least one feature in* $q(o)$*, intersecting the window* w*;*[3]

- *if* o *is a map type and* p *is an instance privilege,* $ext(a) = \{o' | o' \in q(o),\ o' \in s(p),\ o'\ Intersect\ w\}$*, i.e., the extension coincides with the subset of objects in* $q(o)$ *and in the scope of* p*, intersecting* w*;*

- *if* o *is a group object and* p *is an insertion privilege,* $ext(a) = \{o\}$. □

Given a set of authorizations, we assume that two properties are satisfied. The first, *minimality*, imposes that the window and the query are unique for the authorization granted by g to subject s with grant option go, on object o, with privilege p, privilege type pt, and type t. Therefore, if an authorization has to be specified on disjoint regions, the authorization window should be specified as a collection of disjoint polygons, which is an admitted value of $D_{polygons}$. The second, *grant safety*, specifies how the presence of a window constraints the authorizations that can be granted by u. More precisely, authorizations with grant option ($go = true$) can be granted to other users only if $pt = +$ and $go = true$. The window of the granted authorizations must be contained in the corresponding windows of the grantor. As usually done in access control models, we also assume that a special user SA (*Security Administrator*) exists that can grant all possible authorizations, without constraints. Sets of rules satisfying previous properties define a *correct authorization set.*

Definition 6 (Correct authorization set) *Let* $\mathcal{A}$ *be a set of authorizations.* $\mathcal{A}$ *is a correct authorization set (CAS) if the following properties are satisfied:*

- Minimality*:* $\nexists a_1, a_2 \in \mathcal{A}$ *such that* $a_1.u = a_2.u, a_1.o = a_2.o, a_1.p = a_2.p, a_1.pt = a_2.pt, a_1.g = a_2.g, a_1.t = a_2.t, a_1.go = a_2.go$*, and (* $a_1.w \neq a_2.w$ *or* $a_1.q \neq a_2.q$ *);*

- Grant safety*: Let* $a_1 \in \mathcal{A}$*,* $a_1.pt = +$*,* $a_1.go = true$*. Let* $q_{a_1} = \cup\{q | a \in \mathcal{A}, a =< a_1.u, a_1.p, +, g, true, a_1.o, a_1.t, w, q >\}$[4] *and* $w_{a_1} = \cup_{a \in \mathcal{A}}\{w | a \in \mathcal{A}, a =< a_1.u, a_1.p, +, g, true, a_1.o, a_1.t, w, q >\}$.
  Then, $\forall a_2 \in \mathcal{A}$ *such that* $a_2 =< u, a_1.p, +, a_1.u, go, a_1.o, a_1.t, w, q >$ *we must have* $q \subseteq q_{a_1}$[5] *and (*w *In* w_{a_1} *or* w *Equal* w_{a_1}*), where In and Equal are topological relations of TSDM.* □

Note that verifying whether an authorization set is correct, requires to check all pairs of authorizations, thus, if n is the cardinality of the set, the complexity of the check is n^2.

Example 5 *Consider the authorization sets presented in Example 4. The authorization set* $A \cup B$ *is a CAS. Indeed, minimality is obviously satisfied. Moreover,* a_8 *has been granted by BOB to TED from*

[2] Note that when there are different choices when converting a privilege over a feature in a privilege over a map object, the less restrictive choice is chosen. For example, $delete_F$ is converted into $delete_M(2)$, which is less restrictive than $delete_M(1)$ and $delete_M(0)$ (see Section 5.1.2).

[3] $Intersect$ is equivalent to $Touch \vee Overlap \vee In \vee Contains$, and represents the not empty intersection condition.

[4] $\cup$ on $\{q_1, ..., q_n\}$ produces as result the query expression $q_1 \cup ... \cup q_n$.

[5] A query q_1 is contained in another query q_2 if for any database D, $q_1(D) \subseteq q_2(D)$.

authorization a_3, queries in a_3 and a_8 are not specified and $Milan\_City$ In $Milan\_MetropolitanArea$. Additionally, a_9 has been granted by BOB to TED from authorization a_4, the query in a_9 is a refinement of the query in a_4, and the window in a_9 is obviously contained in the window in a_4, which coincides with the overall space. Thus, grant safety is satisfied. On the other hand, $A \cup C$ does not represent a CAS since, although BOB has the grant option on a_3, the condition on windows is not satisfied: $\top$ In $Milan\_MetropolitanArea$ is not true. □

5 Authorization control mechanism

Given a CAS $\mathcal{A}$, the aim of the authorization control mechanism is to determine whether to grant or reject an access request according to what has been specified in $\mathcal{A}$. In general, an access request is a triple $< u, p, o >$ stating that a user u wants to exercise privilege p on object o. In order to check the request, we must consider which other authorizations can be derived from the ones in $\mathcal{A}$. This is achieved by defining specific derivation rules.

In the following, we first present derivation rules and we formally define the authorization base. Then, we show how the authorization base can be used to answer access requests.

5.1 Derivation rules

Given a CAS $\mathcal{A}$, other authorizations can be derived from those in $\mathcal{A}$. Derivation between authorizations can be specified by using *derivation rules*. Given an authorization having a certain form, each derivation rule specifies which other authorizations are implied by it. When this happens, we say that authorization a_2 is derived from authorization a_1, denoted by $a_2 \leftarrow a_1$.

Two different groups of derivation rules can be identified, depending on whether the derivation considers relationships between objects or relationships between privileges. In the following, both groups of authorizations are described.

5.1.1 Derivation over object relationships

Among the objects defined in Section 3.1, it is possible to define some relationships such as if an authorization a exists for object o_1 and there exists a relationship between o_1 and another object o_2, a is propagated from o_1 to o_2. The considered relationships between objects are graphically represented in Figure 3. Authorizations should be propagated from group objects to their instances. Moreover, because of the logical binding between a feature and its spatial representation in one or more map objects inside maps, authorizations for map objects can be derived from authorizations for features. This can be useful, for instance, when we want to grant a user the privilege to access the spatial representation of the instances of a certain feature type.

Next rule specifies derivation between group objects and their instances.

Rule 1 *Let $o \in O$ be a group object. Let $a = \langle u, p, pt, g, go, o, t, w, q \rangle$ be an authorization such that p is an instance privilege. Then, for each $o' \in ext(a)$ the following rule holds:*
$\langle u, p, pt, g, go, o', t, \top, \bot \rangle \leftarrow \langle u, p, pt, g, go, o, t, w, q \rangle$.

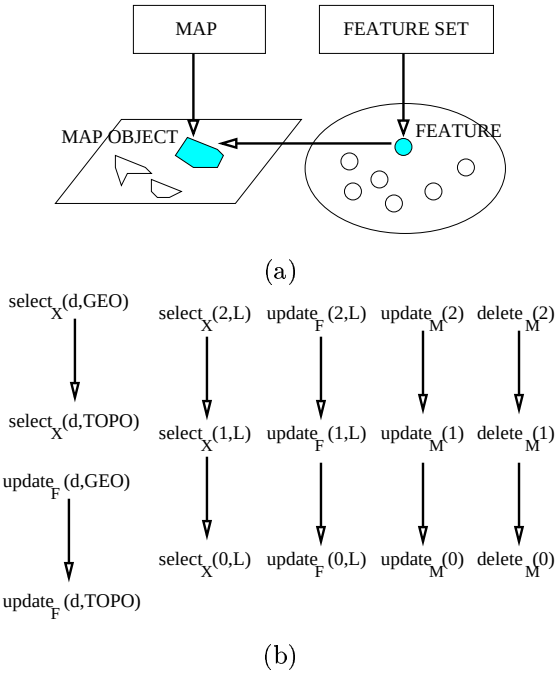

Figure 3: Relationships between (a) objects and (b) privileges

Next rule specifies derivation between features and map objects. This derivation applies only to privileges that make sense for map objects, i.e., those involving spatial representations.

Rule 2 *Let $a = \langle u, p, pt, g, go, f, t, \top, \bot \rangle$ be an authorization such that p is a spatial instance predicate and f is a feature. Then, for each $o' \in ext(a)$, the following rule holds:*

$\langle u, map(p), pt, g, go, o', t, \top, \bot \rangle \leftarrow \langle u, p, pt, g, go, f, t, \top, \bot \rangle$

where function map generates the privilege corresponding to p over map objects (see Definition 5).

Example 6 *Let $County(Milan)$ a feature of County type and $mo = Lomb\_Admin(County(Milan))$ the map object inside the map Lomb_Admin representing the spatial extension of the feature $County(Milan)$. The following are some examples of derivation rules over object relationships:*

$\langle B, select_F(2, GEO), +, A, true, County(Milan), st, \top, \bot \rangle \leftarrow \langle B, select_F(2, GEO), +, A, true, County, st, \top, \bot \rangle$

$\langle B, select_M(2, GEO), +, A, true, mo, st, \top, \bot \rangle \leftarrow \langle B, select_F(2, GEO), +, A, true, County(Milan), st, \top, \bot \rangle$

According to the previous rules, a select privilege $select_F(2, GEO)$ on the feature type County is propagated, due to Rule 1, from County to the feature $County(Milan)$ and, due to Rule 2, from $select_F(2, GEO)$ on the feature $County(Milan)$ to the map object $Lomb\_Admin(County(Milan))$ as $select_M(2, GEO)$. □

5.1.2 Derivation over privilege relationships

Due to the nature of spatial objects, additional derivation rules can be defined in order to take into account object dimension and spatial layer. Such information are contained in the granted privileges. The most informative layer is certainly the geometric one, since topological information can be computed

from it but the converse is not true. Thus, it seems reasonable to assume that an authorization granting a privilege for the geometrical layer has to be propagated to the topological layer. On the other hand, an authorization denying a privilege for the topological layer has to be propagated to the geometrical one.

Similar rules can be defined by considering object dimension. Indeed, an authorization granting a privilege to objects with a certain dimension has to be propagated to objects with lower dimension (e.g., if an user can select regions, he can also select lines and points). On the other hand, an authorization denying a privilege to objects with a certain dimension has to be propagated to objects with higher dimension (e.g., if a user cannot select points, it cannot select neither lines nor regions).

Based on these considerations, it is possible to define a partial order $<$ between privileges, as pointed out in Figure 3(b) ($<$ is represented as $\leftarrow$ in the figure). Next rule formally specifies derivations based on such privilege ordering. Note that the rule can be applied only to objects belonging to the scope of the derived privilege.

Rule 3 *Let $p_1 \in P, p_2 \in P$ such that $p_1 < p_2$ and $o \in s(p_1) \cap s(p_2)$. The following rules hold:*
$\langle u, p_1, +, g, go, o, t, w, q\rangle \leftarrow \langle u, p_2, +, g, go, o, t, w, q\rangle$
$\langle u, p_2, -, g, go, o, t, w, q\rangle \leftarrow \langle u, p_1, -, g, go, o, t, w, q\rangle$.

Example 7 *The following are some examples of derivation rules over privilege relationships:*
$a_1 = \langle B, select_M(2, TOPO), +, A, true, mo, st, \top, \bot\rangle \leftarrow \langle B, select_M(2, GEO), +, A, true, mo, st, \top, \bot\rangle$
$a_2 = \langle B, select_M(1, GEO), +, A, true, mo, st, \top, \bot\rangle \leftarrow \langle B, select_M(2, GEO), +, A, true, mo, st, \top, \bot\rangle$
$a_3 = \langle B, select_M(1, GEO), -, A, true, mo, st, \top, \bot\rangle \leftarrow \langle B, select_M(1, TOPO), -, A, true, mo, st, \top, \bot\rangle$
$a_4 = \langle B, select_M(2, TOPO), -, A, true, mo, st, \top, \bot\rangle \leftarrow \langle B, select_M(1, TOPO), -, A, true, mo, st, \top, \bot\rangle$
According to the previous derivation rules and to Rule 3, a select privilege $select_M(2, GEO)$ on the map object mo is propagated as $select_M(2, TOPO)$ (authorization a_1) and as $select_M(1, GEO)$ (authorization a_2). Moreover, the privilege $select_M(1, TOPO)$ is propagated as $select_M(1, GEO)$ (authorization a_3) and as $select_M(2, TOPO)$ (authorization a_4). □

5.2 Algorithms for access control

Given a CAS $\mathcal{A}$, based on the derivation rules presented above, we can now define the authorization base as the set of authorizations contained in $\mathcal{A}$ extended with those derived from $\mathcal{A}$. The construction of the authorization base guarantees that it represents a correct authorization set.

Definition 7 (Authorization Base) *Let $\mathcal{A}$ be an CAS. Let $\mathcal{A}^+$ be defined as $\{a | a \in \mathcal{A}$ or $\exists a' \in \mathcal{A}^+$ $a \leftarrow a'\}$. The authorization base for $\mathcal{A}$, denoted by $AB(\mathcal{A})$ is defined as $\{\langle u, p, pt, g, go, o, t, w, q\rangle \mid \langle u, p, pt, g, go, o, t, w', q'\rangle \in \mathcal{A}^+$, $w = \cup\{w_i | \langle u, p, pt, g, go, o, t, w_i, q_i\rangle \in \mathcal{A}^+\}$, $q = \cup\{q_i | \langle u, p, pt, g, go, o, t, w_i, q_i\rangle \in \mathcal{A}^+\}$ $\}$.* □

Proposition 1 *Let $\mathcal{A}$ be an CAS. Then, $AB(\mathcal{A})$ is a CAS.*

Proof Sketch: By construction, minimality is satisfied by $AB(\mathcal{A})$. Note that minimality may not be satisfied by $\mathcal{A}^+$. This happens for example when $\mathcal{A}$ contains an authorization with privilege $p(2, l)$ and an authorization with privilege $p(1, l)$. When applying derivation rules, a new authorization with

privilege $p(1,l)$ is generated that may violate minimality. Grant safety can be proved by contradiction by observing that derivation rules do not change users, grantors, grant option, authorization sign and type, queries and windows. Thus, if the second property is not satisfied by the authorization base, it cannot be satisfied by $\mathcal{A}$ but this contradicts the hypothesis. □

The following proposition gives an estimated of the complexity of computing the authorization base and of its cardinality.

Proposition 2 *Let $\mathcal{A}$ be a CAS and $AB(\mathcal{A})$ the corresponding authorization base. Let n_a be the number of authorizations in $\mathcal{A}$. Let n_m be the number of maps in the database and n_o be the maximal cardinality of group objects extensions. Assuming that $n_m <<< n_o$, the cardinality of $AB(\mathcal{A})$ is linear in n_a and n_o. The complexity of constructing $AB(\mathcal{A})$ is quadratic in n_a and n_o.*

Proof Sketch: The proof follows from the following considerations: (i) The longest derivation can be obtained by considering authorizations on group objects or features (otherwise Rules 1 and 2 are not used). (ii) Given an authorization a such that $a.o$ is a group object or a feature, the longest derivation starting from a is obtained by first applying Rule 3 or 4 to a. According to Figure 3, this can be done at most 3 times. Then, if $a.o$ is a group object, Rule 1 can be applied once for each object belonging to the extension of $a.o$, which is lower than or equal to n_o. If $a.o$ is a feature, Rule 2 can be applied at most once for each map object corresponding to $a.o$. Since maps are at most n_m, the number of map objects is lower than or equal to n_m. (iii) From the previous consideration, since we assume that $n_m <<< n_o$, it follows that from each authorization we can generate at most $3n_o$ authorizations. Thus, from $\mathcal{A}$, we can generate at most $3n_on_a$ authorizations. Thus, the cardinality of $AB(\mathcal{A})$ is linear in n_a and n_o. (iv) On the other hand, in order to construct $AB(\mathcal{A})$, windows have to be combined, thus, each authorization in the set has to be compared with all the others. Thus, the complexity is quadratic in n_a and n_o. □

Example 8 *Consider the geographical database described in Example 1. Let Bob be an user, having only the following two authorizations:*

$a =< BOB, select_M(2, GEO), +, ANN, true, Lomb\_rail, st, \top, \bot >$

$b =< BOB, select_F(1, TOPO), -, ANN, true, Railway, st, \top, \bot >$

In Lomb_rail, Railway instances have dimension 1, Accident instances have dimension 0, no object with dimension 2 exists. Moreover, both Railway and Accident features admit both a geometric and topological representation. Thus, from a, we can derive the following authorizations:

$a_1 =< BOB, select_M(1, GEO), +, ANN, true, Lomb\_rail, st, \top, \bot >$, *by applying Rule 3 to a.*

$a_2 =< BOB, select_M(0, GEO), +, ANN, true, Lomb\_rail, st, \top, \bot >$, *by applying Rule 3 to a or to* a_1.

$a_3 =< BOB, select_M(2, TOPO), +, ANN, true, Lomb\_rail, st, \top, \bot >$, *by applying Rule 3 to a.*

$a_4 =< BOB, select_M(1, TOPO), +, ANN, true, Lomb\_rail, st, \top, \bot >$, *by applying Rule 3 to* a_3.

$a_5 =< BOB, select_M(0, TOPO), +, ANN, true, Lomb\_rail, st, \top, \bot >$, *by applying Rule 3 to* a_4 *or* a_3.

$a_1^i =< BOB, select_M(1, GEO), +, ANN, true, x^i_{rail}, st, \top, \bot >$, *where* $x^i_{rail} \in \{y | y \in Lomb\_rail \wedge y$ *represents a Railway feature* $\}$; *these authorizations are obtained by applying Rule 1 to* a_1.

$a_2^i =< BOB, select_M(0, GEO), +, ANN, true, x^i_{acc}, st, \top, \bot >$, *where* $x^i_{acc} \in \{y | y \in Lomb\_rail \wedge y$ *represents an Accident feature* $\}$; *these authorizations are obtained by applying Rule 1 to* a_2.

$a_4^i =< BOB, select_M(1, TOPO), +, ANN, true, x_{rail}^i, st, \top, \bot >$, *where* $x_{rail}^i \in \{y | y \in Lomb\_rail \wedge y$ *represents a Railway feature* $\}$*; these authorizations are obtained by applying Rule 1 to* a_4.
$a_5^i =< BOB, select_M(0, TOPO), +, ANN, true, x_{acc}^i, st, \top, \bot >$, *where* $x_{acc}^i \in \{y | y \in Lomb\_rail \wedge y$ *represents an Accident feature* $\}$*; these authorizations are obtained by applying Rule 1 to* a_5.

From b*, we get the following authorizations:*
$b_1 =< BOB, select_F(2, TOPO), -, ANN, true, Railway, st, \top, \bot >$, *by applying Rule 3 to* b.
$b_2 =< BOB, select_F(1, GEO), -, ANN, true, Railway, st, \top, \bot >$, *by applying Rule 3 to* b.
$b_3 =< BOB, select_F(2, GEO), -, ANN, true, Railway, st, \top, \bot >$, *by applying Rule 3 to* b_2 *or* b_1.
$b_2^i =< BOB, select_F(1, GEO), -, ANN, true, x_{rail}^i, st, \top, \bot >$, *where* x_{rail}^i *represents a Railway feature; these authorizations are obtained by applying Rule 1 to* b_2.
$b^i =< BOB, select_F(1, TOPO), -, ANN, true, x_{rail}^i, st, \top, \bot >$, *where* x_{rail}^i *represents a Railway feature; these authorizations are obtained by applying Rule 1 to* b.
$b_2^{i,j} =< BOB, select_M(1, GEO), -, ANN, true, x_{rail}^{i,j}, st, \top, \bot >$, *where* $x_{rail}^{i,j} \in \{y | y \in Lomb\_rail \wedge y$ *represents a Railway feature* $\}$*; these authorizations are obtained by applying Rule 2 to* b_2.
$b^{i,j} =< BOB, select_M(1, TOPO), -, ANN, true, x_{rail}^{i,j}, st, \top, \bot >$, *where* $x_{rail}^{i,j} \in \{y | y \in Lomb\_rail \wedge y$ *represents a Railway feature* $\}$*; these authorizations are obtained by applying Rule 2 to* b. □

Given an access request $r =< u, p, o >$, stating that a user u wants to exercise privilege p on object o, and a *CAS* $\mathcal{A}$, the problem arises of establishing whether the request can or cannot be satisfied and on which set of objects. To this purpose, the authorization base is used. More precisely, given an authorization a, we say that $r =< u, p, o >$ depends on a if and only if $a = \langle u, p, pt, g, go, o, t, w, q \rangle$, i.e., a grants or deny privilege p to u over object o. According to what we presented in Section 4, the access request can be satisfied if and only if the following conditions are satisfied:

1. r depends on a strong positive authorization and on no strong negative authorizations;
2. r depends on a weak positive authorization, on no weak negative authorizations, and on no strong authorizations.

In all the other cases, the access has to be denied. This means that, in presence of strong authorizations, weak ones are not considered. For both weak and strong authorizations, negative authorizations have precedence with respect to positive ones.

Definition 8 *Let* $\mathcal{A}$ *be a CAS and* $r =< u, p, o >$ *be an access request.* r *is satisfied in* $\mathcal{A}$ *if and only if one of the following condition holds:*

- $\exists \langle u, p, +, g, go, o, st, w, q \rangle \in AB(\mathcal{A})$ *and* $\nexists \langle u, p, -, g, go, o, st, w, q \rangle \in AB(\mathcal{A})$;
- $\exists \langle u, p, +, g, go, o, wk, w, q \rangle \in AB(\mathcal{A})$, $\nexists \langle u, p, -, g, go, o, wk, w, q \rangle \in AB(\mathcal{A})$ *and* $\nexists \langle u, p, pt, g, go, o, st, w, q \rangle \in AB(\mathcal{A})$.

If r *is satisfied in* $\mathcal{A}$*, we define* $O(r) = \{o' | o'$ *is an instance of* o *and* $< u, p, o' >$ *is satisfied in* $\mathcal{A}\}$ *as the set of instance objects over which the privilege is granted.* □

The complexity of checking an access request depends on the cardinality of the authorization base. Thus, it is linear in n_a and n_o (see Proposition 2).

Example 9 *Consider the authorization base computed in Example 8. Suppose that BOB makes the following access request:* $< BOB, select_M(1, GEO), Lomb\_rail >$. *Since authorizations* $b^{i,j}$ *and* a_1^i, *as well as* $b_2^{i,j}$ *and* b_4^i *are all strong but conflicting, and since we give precedence to negation, Lomb_rail map objects corresponding to Railway features cannot be accessed. On the other hand, map objects corresponding to Accident features can be accessed both at the geometrical and topological layer, due to authorizations* a_5^i *ad* a_2^i. *Now suppose that authorization b (and therefore all the authorizations derived from b) are weak. In this case, since strong positive authorizations exist for Railway and Accident map objects, all of them can be accessed both at the geometrical and topological layer. As a third case, suppose that authorization a (and therefore all the authorizations derived from a) are weak. Independently on whether authorization b is weak or strong, only Accident map objects can be accessed.* □

6 Concluding remarks

In this paper we have presented a new access control model for geographical maps. Our model supports both positive and negative authorizations. Moreover, by means of strong and weak authorization concepts, our model permits inheritance of the authorizations according to the objects hierarchy and propagation of authorizations, taking into account object dimension and type of spatial information.

It is interesting to note that, based on Definition 4, a window assigns an authorization a spatial extent in the same space in which the map objects of the application are located. However the authorization consists also of properties which can be simply expressed by alphanumeric descriptions. An authorization thus can be considered itself an entity with spatial and non spatial properties and as such it can be naturally modeled as a TSDM feature with a specific feature type. Moreover, the map in which the spatial extents of authorizations (i.e. the windows) are represented corresponds to an authorization map, that is structurally analogous to the maps of the geographical database. As a result, a uniform model is applied for both representing application data and authorizations and the same operations applicable to feature type extensions can be applied to authorizations as well. For example, one can formulate a map selection query to get all the authorizations granted on a given region.

As part of our future work, we are interested in defining efficient techniques for security administration. To this purpose, we plan to use the authorization map to perform administrative operations. Moreover, we plan to define more efficient techniques for checking access requests. Indeed, the proposed approach is very inefficient since the overall authorization base is computed even if only a small subset of it is in general useful to answer the request. To this purpose, we are currently investigating how logic programming techniques [11] can be used to represent authorization sets, to construct the authorization base and to check user requests. An additional issue we plan to investigate concerns the development of similar access control models for GIS standards [9, 10].

References

[1] V. Atluri and P. Mazzoleni. A Uniform Indexing Scheme for Geo-Spatial Data and Authorizations. In *Proc. of the Sixteen Conf. on Data and Application Security*, 2002.

[2] A. Belussi, B. Catania, and E. Bertino. A Reference Framework for Integrating Multiple Representations of Geographical Maps. In *Proc. of ACM GIS*, 2003.

[3] E. Bertino, M.L. Damiani, D. Momini. An Access Control System for a Web Map Management Service. In *Proc. of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE-WS-ECEG 2004)*, 2004.

[4] E. Bertino and M.L. Damiani. A Controlled Access to Spatial Data on Web. In *Proc. of the 7th AGILE Conference on Geographic Information Science.* April 2004.

[5] E. Bertino, S. Jojodia, and P. Samarati. Supporting Multiple Access Control Policies in Database Systems. In *IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Oakland, CA 1996.

[6] S. A. Chun and V. Atluri. Protecting Privacy from Continuous High-resolution Satellite Surveillance. In *Proc IFIP Workshop on Database Security*, pages 233-244, 2000.

[7] E. Clementini, P. di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *LNCS 692: Proc. of SSD'93*, pages 277–295, 1993.

[8] M. J. Egenhofer. Reasoning about Binary Topological Relations. In *LNCS 525: Proc. of SSD'91*, pages 143–160, 1991.

[9] ISO TC 211 Geographic information/Geomatics. 19109, Geographic information - Rules for Application Schema *text for FDIS, doc. N. 1538*, 2003.

[10] ISO TC 211 Geographic information/Geomatics. 19107, Geographic information - Spatial Schema. *text for FDIS, doc. N. 1324*, 2002.

[11] J.W. Lloyd. *Foundations of Logic Programming*. 2nd ed. Berlin:Springer-Verlag, 1987.

[12] OpenGIS Consortium, OpenGIS Simple Features Specification for SQL, *OGC 99-049*, 1999.

[13] E. Puppo and G. Dettori. Towards a Formal Method for Multiresolution Spatial Maps. In *LNCS 951: Proc. of SSD'95*, pages 152-169, 1995.

[14] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. Supporting Multiple Access Control Policies in Database Systems. *ACM Transactions on Database Systems*, Vol 16, No 1, pages 88-131, 1991.

[15] P. Rigaux, M. Scholl, and A. Voisard. Spatial Databases with Application to GIS. Morgan Kaufmann, 2002.

[16] H. Shen and P. Dewan. Access Control for Collaborative Environments In *Proc Int'l Conference of Computer Supported Cooperative Work*, pages 51-58, 1992.