CERIAS Tech Report 2006-05

GEO-RBAC: A SPATIALLY AWARE RBAC

by Elisa Bertino, Barbara Catani, Maria Damiani, Paolo Perlasca

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

GEO-RBAC: A Spatially Aware RBAC

Elisa Bertino
Purdue University, West Lafayette, USA
Barbara Catania
University of Genoa, Italy
Maria Luisa Damiani
University of Milan, Italy & EPFL Lausanne, CH
and
Paolo Perlasca
University of Milan, Italy

Securing access to data in location-based services and mobile applications requires the definition of spatially aware access control systems. Even if some approaches have already been proposed either in the context of geographic database systems or context-aware applications, a comprehensive framework, general and flexible enough to cope with spatial aspects in real mobile applications, is still missing. In this paper, we make one step towards this direction and we present GEO-RBAC, an extension of the RBAC model to deal with spatial and location-based information. In GEO-RBAC, spatial entities are used to model objects, user positions, and geographically bounded roles. Roles are activated based on the position of the user. Besides a physical position, obtained from a given mobile terminal or a cellular phone, users are also assigned a logical and device independent position, representing the feature (the road, the town, the region) in which they are located. To make the model more flexible and re-usable, we also introduce the concept of role schema, specifying the name of the role as well as the type of the role spatial boundary and the granularity of the logical position. We then extend GEO-RBAC to cope with hierarchies, modeling permission, user, and activation inheritance, and separation of duty constraints. The proposed classes of constraints extend traditional ones to deal with different granularities (schema/instance level) and spatial information. They represent an attempt to define a suitable class of constraints for spatially-aware applications. The paper is concluded with the investigation of several properties concerning the resulting model.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Spatial Databases and GIS*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Management, Security, Theory
Additional Key Words and Phrases: GIS, Authorization model, Access control, Location-based services

Author's address: E. Bertino, CERIAS, CS & ECE Department, Purdue University, West Lafayette, USA, E-mail:bertino@cerias.purdue.edu
B. Catania, Dipartimento di Informatica e Scienze dell'Informazione, Università degli Studi di Genova, via Dodecaneso 35, 16146 Genova, Italy, E-mail: catania@disi.unige.it
M. L. Damiani, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, via Comelico 39/41, 20135 Milano, Italy; EPFL, Lausanne, CH, E-mail: damiani@dico.unimi.it.
P. Perlasca, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, via Comelico 39/41, 20135 Milano, Italy, E-mail: perlasca@dico.unimi.it.

1. INTRODUCTION

The widespread deployment of location-based services and mobile applications as well as the increased concern for the management and sharing of geographical information in strategic applications like environmental protection and homeland security have resulted in a strong demand for spatially aware access control systems. These application domains pose interesting requirements against access control systems. In particular, the permissions assigned to users depend on their position in a reference space; users often belong to well defined categories; objects to which permissions must be granted are located in that space; access control policies must grant permissions based on object locations and user positions.

As an example consider a mobile application for the personnel and patients of a health care organization. Individuals are given a location-aware terminal with which they can request information services provided by an application server. The organization consists of individuals who have different functional roles, e.g. Nurse, Doctor and Patient. We can notice that, depending of the organizational context, the services available to users may differ based on the functional roles of users. For example, the services available to nurses may be different from those available to doctors, not simply because of the individual preferences but mainly because of organizational and functional reasons. Further, the availability of services may depend on the position of the requester. For example, a nurse may be allowed to request the record of a patient only when located in the department she has been assigned to. To deal with the requirements listed above, an access control model with spatial capabilities is needed. Since in location-aware applications users are often grouped in distinct categories, like *nurse* and *doctor* role-based access control models (RBAC models) [Ferraiolo et al. 2001; Sandhu et al. 2000] represent a reasonable choice. Various role-based and spatially aware access control systems have been proposed for securing access to spatial data stored in a spatial DBMS or for securing access to location-aware applications. Even though some preliminary proposals have been reported adding contextual information, such as spatial and temporal information to access control mechanisms, such approaches are simplistic and do not account for several of the requirements we have devised such as multigranularity of position and relationships in space.

In this paper, we overcome those limitations by proposing a comprehensive spatial framework for an access control system securing access to spatial data in location-aware applications. Such a model, called GEO-RBAC, extends the RBAC model with the concept of *spatial role* and supports the homogeneous representation of all spatial aspects involving roles, objects and contextual information such as user position. The spatial model we adopt is compliant with OGC (Open GeoSpatial Consortium) [Open GIS Consortium 1999]. Thus, it is based on the notion of feature type (a road, a town, a region) and feature, as instance of a given feature type (road A10, Milan, Lombardy). Features have a well defined geometry (representing points, lines, or polygons) in a reference space. Objects in GEO-RBAC correspond to sets of features of a given type.

A spatial role in GEO-RBAC represents a geographically bounded organizational function. The boundary is defined as a feature, such as a road, a city or a hospital. The boundary specifies the spatial extent in which the user is to be located for being enabled to play such a role. Besides a physical position, obtained from a given mobile terminal such as a GPS

based vehicle tracking device or a cellular phone, users are also assigned a logical and device independent position, representing the feature in which the user is located. *Logical positions* can be computed from *real positions* by using specific mapping functions. To make the model more flexible, we assume that logical positions can be represented at different granularities, depending on the spatial role played by the user. If the user is located inside the spatial boundary of the role which has been selected (*activated*) during the session she has logged in, the role is said to be *enabled*. To specify the type of the spatial boundary of the role and the granularity of the logical position, we introduce the concept of *spatial role schema*. Spatial roles are thus specified as instances of role schemas. The usage of role schemas and instances makes our model quite flexible since the type of role extents and logical positions can be customized (and the definition re-used), depending on the function the role represents.

GEO-RBAC is a comprehensive model, which like RBAC, consists of three components referred to Core, Hierarchical and Constrained GEO-RBAC, respectively, which are presented in this paper. The contributions of our work can be summarized as follows:

—Core GEO-RBAC specifies the basic concepts of the model, thus the notion of spatial role, role schema, real/logical position, activated/enabled role, which are used by the subsequent components.

—Hierarchical GEO-RBAC extends the conventional concept of hierarchy by introducing two major novelties. First, two distinct hierarchies are provided, one over role schemas and one over role instances. The role schema hierarchy supports the inheritance of permissions and user memberships among sets of homogeneous roles and thus further simplifies role definition. The second extension concerns the formal definition of role activation and enabling in the presence of hierarchies. To this purpose, we present a model in which the role instance hierarchy is used to derive the roles which not only are activated but also enabled in a session.

—Constrained GEO-RBAC supports the specification of separation of duty (SoD) constraints for spatial roles and role schemas. Since exclusive role constraints are important to support the definition and maintenance of access control policies in mobile contexts, SoD constraints are extended to account for different granularities (schema/instance level), dimension (spatial/non-spatial), and different verification time (static, dynamic at activation time, dynamic at enabling time). The resulting set of constraints represents a first comprehensive class of constraints for spatially-aware applications.

—Properties of Constrained GEO-RBAC. Even if the investigation of administrative operations for GEO-RBAC is outside the goals of this paper, an analysis on the expressivity and the complexity of the proposed constraints is a relevant issue in order to establish the usability of the proposed model. Some of such properties extend already known results to the new classes of constraints we have introduced. Other properties are new and account for the specific characteristics GEO-HRBAC .

The remainder of this paper is organized as follows. In Section 2, we discuss related work. The reference geometric model we consider in this paper and its usage in GEO-RBAC are introduced in Section 3. In Section 4, we present the core model of GEO-RBAC whereas hierarchies are discussed in Section 5. An overall example is then presented in Section 6, which is then used in Section 7 to discuss the proposed classes of constraints. Properties of the resulting model are then discussed in Section 8. Finally, Section 9 presents

some concluding remarks and outlines future work. The Appendix contains proofs of results.

2. RELATED WORK

Related research spans across several fields, from security to GIS (Geographical Information Systems) to context-based processing and mobile applications. The basis of our model is the role-based access control. Since the seminal paper of Sandhu et al. [1996], the RBAC model has gained increasingly consensus in the research community as well as in industry to finally become a standard widely adopted by organizations.

By contrast, the concern for the integration of the spatial dimension into RBAC-based models has emerged only recently as a consequence of the growing relevance of geo-spatial information in advanced GIS and mobile applications. In GIS, the demand for spatially aware access control systems is primarily motivated by the need of sharing geographical information across local and national boundaries. To our knowledge, the first access control model for geographical data has been proposed in [Atluri and Mazzoleni 2002; Chun and Atluri 2000] to control the access to satellite image maps. An access control system for geometric and vector-based spatial data has been proposed in [Bertino et al. 2004]. The model introduces the concept of spatial authorization as an authorization that can be defined only on portions of space. When an access request is made for an object, the system checks whether the requested object lies in the authorization space and if this is the case, it grants the access. This model has been applied to support controlled access to spatial data on Web. The underlying spatial data model is, however, relatively simple and does not address important issues such as the multi-granularity of spatial data. A similar architecture, but focused on XML-based representation of spatial data, has been proposed in [Purevjii et al. 2004]. A more complex spatial data model has been assumed in [Belussi et al. 2004]. In this work, an access control system is presented that allows the specification of authorization rules to access complex structured spatial data stored in a DBMS and organized according to multiple spatial representation levels and at multiple granularities. The system, however, does not deal with geographically bounded roles neither with mobile users. An approach which integrates geo-spatial and security standards to support controlled access to spatial information through geo-Web services is presented in [Matheus 2005]. In this work, a policy specification language GeoXACML is defined as a geo-spatial extension of the OASIS standard eXtensible Access Control Markup Language (XACML). GeoXACML allows the specification of rules which enable or deny the access to geo-spatial objects based on spatial criteria, such as containment relationships. None of these models however is conceived for use in a dynamic environment, which instead is the main concern of spatial and non-spatial context aware access control models.

Non-spatial context-aware access control models include Generalized TRBAC (GTRBAC) [Joshi et al. 2005] which incorporates a set of language constructs for the specification of various temporal constraints on roles, including constraints on role enabling, role activation, user-to-role assignments, and permission-to-role assignments. We borrow from this model the distinction between role enabling and activation. X-GTRBAC [Bhatti et al. 2005] augments GTRBAC with XML for supporting the policy enforcement in a heterogeneous, distributed environment. In addition to temporal constraints, the model also supports non-temporal contextual constraints. The approach, however, is more focused on the software engineering aspects of the access control rather than on the expressivity of the

policy specification language. A notable approach is the one proposed through the Generalized RBAC (GRBAC) [Covington et al. 2001; Covington et al. 2000]. GRBAC introduces the concept of environment roles, that is roles that can be activated based on the value of conditions in the environment where the request has been made. Environmental conditions include time, location, and other contextual information that is relevant to access control. If compared with GEO-RBAC, the concepts of role extent and user position are close to that of context variables. However, the mechanism of contexts is very general and does not account for the specificity of spatial information, such as the multi-granularity of position and the spatial relationships that may exist between the spatial elements in space. Moreover, in GEO-RBAC a common spatial data model is adopted in order to provide a uniform and standard based representation of locational aspects that, notably, involve not only roles but also protected objects. The spatial dimension of access control is the basic ingredient of the approach presented in [Hansen and Oleshchuk 2003a; 2003b]. In such work, an extension of RBAC model is proposed based on the notion of spatial role, intended as a role that is automatically activated when the user is in a given position. The space model is however very simple and targeted to wireless network applications. It consists of a set of adjacent cells and the position of the user is the cell or the aggregate of cells containing it. The spatial granularity of the position is thus fixed while the space is rigidly structured and the position itself does not have any semantic meaning but simply a geometric value. By contrast, in our model the granularity of the user position may depend on the role of the user; thus no assumption is made on the space layout. Moreover, the spatial dimension integrates geometric and semantic knowledge about the world. A different approach which combines space and time is presented in [Chandran and Joshi 2005]. Such system borrows from GEO-RBAC the distinction between real position and logical position and from GTRBAC the notion of temporal context. Though, the model does not include the notion of schema, neither supports important features of GEO-RBAC such as hierarchies of enabled roles and spatially-aware separation of duty constraints.

3. SPATIAL INFORMATION IN GEO-RBAC

In order to make RBAC spatially aware, we need to first introduce the reference geometric model we want to use. In GEO-RBAC, the geometric model is used to represent objects, to model user positions, and to assign spatial extents to roles.

3.1 The reference geometric model

The geometric model describes how locations on Earth are represented in GEO-RBAC. We assume objects to be embedded in the Euclidean space E whilst a spatial reference system maps locations in E onto places on Earth. We assume objects to have a geometric representation (geometry) compliant with the OGC (Open GeoSpatial Consortium) *simple feature* geometric model [Open GIS Consortium 1999]. We adopt this model because it is widely deployed in commercial spatial DBMSs and GISs. Although a more advanced spatial data model has been recently proposed [Open GIS Consortium 2001; 2003], we do not loose in generality by adopting the simple feature model.

In such a model, the geometry of an object can be of type point, line or polygon, or recursively be a collection of disjoint geometries. A point describes a single location in the coordinate space; a line represents a linear interpolation of an ordered sequence of points; a polygon is defined as an ordered sequence of closed lines defining the exterior and interior boundaries of an area. An interior boundary defines a hole in the polygon.

In GEO-RBAC, we consider the set of all geometries contained in a reference space (a polygon) and we denote it with GEO. We denote with MBB the reference space. Geometries can be related by different types of relationship. Among them, the reference set of topological relations is $REL = \{Disjoint, Touch, In, Contains, Equal, Cross, Overlap\}$. 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 by Clementini et al. [1993]. To exemplify, the $Contains(x, y)$ relationship between geometries x and y holds when all points of y are also points of x.

3.2 Spatially aware objects

We assume resources to be protected consist of data about entities of the real world that may occupy a position. To be compliant with the OGC terminology, we call these entities *features* [Open GIS Consortium 1999]. Features are identified by names. $Milan$, lake $Michigan$, car identified by $AZ213JW$ are examples of features. Features are *spatial* when entities can be mapped onto locations in the given space (for example, $Milan$ and lake $Michigan$). The location of a feature is represented through a geometry. Conversely, features are *non-spatial* when they are not associated with any location (for example car identified by $AZ213JW$). The sets of spatial features and non-spatial features are denoted in the following respectively by F_s and F_{ns} with $F_s \cap F_{ns} = \emptyset$. We define the set of features $F = F_s \cup F_{ns}$. Feature location is formally defined as follows.

Definition 3.1 *Feature location.* Let F be the set of features and GEO be the set of geometries in space E. Feature location is a function $LocObj : F \rightarrow GEO \cup \{\perp\}$. Given a feature $f \in F$, the location $LocObj(f)$ is either a geometry in GEO if $f \in F_s$ or undefined ($\perp$) if $f \in F_{ns}$. We assume that the dimension of a feature f, denoted by $dim(f)$, is the geometric type of its location: 0 if it is a point, 1 if it is a line, 2 if it is a polygon, and $\perp$ if $f \in F_{ns}$. $\diamond$

Features have an application dependent semantics that is expressed through the concept of *feature type* [Open GIS Consortium 1999]. A feature type captures the intensional meaning of the entity. $Road$, $Town$, $Lake$, Car are examples of feature types. The *extension* of a feature type ft, denoted by $Ext(ft)$, is a set of semantically homogeneous features. We assume, without loosing in generality, that the dimension of a spatial feature type is the dimension of its instances. For example, $Road$ may have dimension 1 whereas $Town$ and $Lake$ may have dimension 2. A feature type is instead non-spatial when the extension only includes non-spatial features (for example, Car). The two sets of spatial feature types and non-spatial feature types are indicated respectively by FT_s and FT_{ns} with $FT_s \cap FT_{ns} = \emptyset$. We define the set of features types $FT = FT_s \cup FT_{ns}$. Next definition introduces some functions relevant for feature management.

Definition 3.2 *Features functions.* Let $FT = \{ft_1, \ldots, ft_n\}$ be the set of feature types, F and GEO the set of features and geometries, respectively. We define:

—$FT\_dim : FT \rightarrow \{0, 1, 2, \perp\}$ such that, given a feature type ft, $FT\_dim(ft) = 0$ if ft is of type point, $FT\_dim(ft) = 1$ if ft is of type line, $FT\_dim(ft) = 2$ if ft is of type polygon, $FT\_dim(ft) = \perp$ when $ft \in FT_{ns}$.

—$Ext : FT \rightarrow 2^F$, the mapping from a feature type, either spatial or non-spatial, to a subset of features such that, given $ft_i \in FT, \forall f \in Ext(ft_i), dim(f) = FT\_dim(ft_i)$. Given a feature type ft_i, $Ext(ft_i)$ represents the *extension* of ft_i.

—$FT\_Type : F \rightarrow FT$ the mapping from a feature to its feature type. ◇

In some application contexts, it may happen that some spatial relationship exists between feature type extensions, defining a partial order between feature types. Consider for example feature types $Region$ and $Town$. It is reasonable to assume that the geometry associated with instances of $Town$ is contained in the geometry of instances of $Region$. If that is the case, we say the the feature type $Town$ is contained in the feature type $Region$. Moreover, we assume that the partial order is a bounded lattice, i.e., we assume FT contains two system-defined feature types: a greatest (top) feature type and a least (bottom) feature type. The *top* feature type, denoted by Top_{ft}, contains all the user-defined feature types. The *bottom* feature type, denoted by Bot_{ft}, is contained in all the user-defined feature types. A we will see, the relationship of containment will be useful in characterizing the relationships between locations and role extents.

Definition 3.3 *Feature type ordering*. The feature type ordering is defined as a bounded lattice $(FT, \subseteq_{ft})$ such that:

(1) FT is a set of feature types containing the system-defined feature types Top_{ft} and Bot_{ft} such that:
   (a) $Ext(Top_{ft}) = \{Top_f\}, LocObj(Top_f) = MBB$
   (b) $Ext(Bot_{ft}) = \{Bot_f\}, LocObj(Bot_f) = \perp$

(2) $\preceq_{ft}$ is a partial order over FT defined as follows: $ft_i \subseteq_{ft} ft_j$ holds if $\forall f_i \in Ext(ft_i)\ \exists f_j \in Ext(ft_j)$ and $LocObj(f_i) \subseteq LocObj(f_j)$.

We notice that $\forall ft_i \in FT, ft_i \subseteq_{ft} Top_{ft}$ and $Bot_{ft} \subseteq_{ft} ft_i$. ◇

In order to more easily assign permissions, we assume that objects in GEO-RBAC are represented as subsets of feature type extensions. Formally, objects are defined as follows.

Definition 3.4 *Objects in GEO-RBAC*. Let FT be the set of feature types. Objects in GEO-RBAC are defined as $OBJ = \bigcup_{ft \in FT} 2^{Ext(ft)}$. Thus, the set OBJ consists of all possible subsets of feature type extensions. ◇

Objects in GEO-RBAC can be extensionally represented by listing the features belonging to the set or by intensionally specifying a query either spatial or non-spatial over a feature type extension. The object in this case corresponds to the query result.

3.3 Spatial role

The central notion in GEO-RBAC is that of *spatial role* defined as a pair $< r, e >$, where r is the role name and e the *spatial extent* (extent for short) of the role. The role extent defines the boundaries of the space in which the role can be assumed by the user.

Moreover, it seems reasonable to assume that the extent of a role, besides a geometry, has a semantic characterization. Thus, we assume role extents to be modeled as features of possibly different feature types. As a further consideration, we note that in real applications it makes sense to have also non-spatial roles. For example, it does not seem reasonable to assign spatial extents to roles related to company organizations such as $Manager$ or $Employee$. However, for the sake of uniformity, we consider non-spatial roles to be a subset of spatial roles having the spatial feature denoting the reference space, i.e. Top_f, as role extent.

Notation	Meaning
FT	Feature types
F	Features
R	Role names
$REXT\_FT$	Feature types of role extents
$LPOS\_FT$	Feature types of logical positions
$REXT$	Role extents
$LPOS$	Logical positions
$RPOS$	Real positions
M	Position mapping functions
OBJ	Objects

Table I. Notation for the main sets used in GEO-RBAC

Definition 3.5 *Role extent*. Let R be a set of role names, let $REXT\_FT \subseteq FT$ be the set of role extent feature types. The set of role extents, denoted by $REXT$, is defined as $REXT = \bigcup_{ft \in REXT\_FT} Ext(ft)$. ◇

Notice that the same role name can appear in different spatial roles. For example the role *Doctor* can be associated with different extents, say different hospitals, to form distinct spatial roles.

3.4 Position Model

In GEO-RBAC, we assume users to have a position that can change in time. Positions can be real or logical. The real position corresponds to the position on the Earth of the user, obtained from a given mobile terminal such as a GPS based vehicle tracking device or a cellular phone. Real positions can be represented as geometries of different types since, depending on the chosen technology and accuracy requirements, they may correspond to points or polygons. For the sake of generality we do not make any assumption on the geometric type of the real position.

Besides real positions, however, for activating a given role, it may be useful to know not only the real position of the user but also the logical one. The logical position allows a position to be represented in a way that is almost independent from the underlying positioning technology. Logical position is modeled as a spatial feature. For example the logical location of a vehicle may be a polygonal feature of type, say, $city$. Such a feature can already exist in the information base or be a new feature entered into the system when the position is notified. Positions can also be represented at varying granularity levels which may depend on the role played by the user: for example for a taxi driver the logical position can be a point along a road while for a truck driver it may be a portion of road. Note that a coarse position may be requested for privacy-preserving purpose, in order to hide the actual position of user.

The logical position can be computed from real positions by using specific mapping functions. For example, a function could be defined to map a point acquired through GPS based equipment onto the closer road segment.

Definition 3.6 *Positions*. The set $RPOS$ of real positions is a subset of geometries in GEO, thus $RPOS \subseteq GEO$. The set $LPOS$ of logical positions consists of features of type in $LPOS\_FT \subseteq FT$, thus $LPOS$ is defined as $LPOS = \bigcup_{ft \in LPOS\_FT} Ext(ft)$.

Given a feature type ft, we call *position mapping function for* ft a function m_{ft} defined

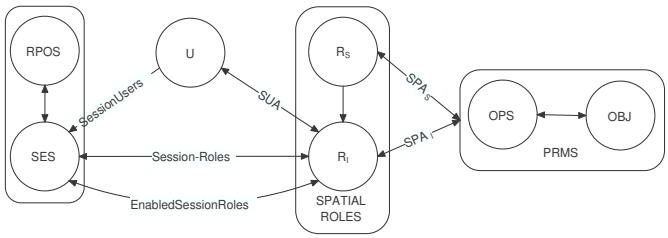

Fig. 1. Core GEO-RBAC

as $m_{ft} : RPOS \rightarrow LPOS$ such that $m_{ft}(rp) = f$ and $f \in Ext(ft)$. The function m_{ft}, given a real position rp, returns a logical position corresponding to an instance of ft having rp as real position. ◇

A position mapping function is a total function, thus the logical position can be computed for any real position. Moreover, we assume to have at least one position mapping function for each feature type ft. Further, we introduce for future use, two constant position mapping functions named as *bottom* and *top*, defined as follows: $m_{bot}(rp) = Bot_f$, $m_{top} = Top_f$. The two functions return respectively the feature corresponding to the whole space and the feature indicating the undefined space. We denote with M the set of all position mapping functions.

4. THE GEO-RBAC CORE MODEL

The central idea of GEO-RBAC is the distinction between the concept of role schema and role instance. A role schema defines some common properties of a set of spatially aware organizational functions with a similar meaning. A role schema not only defines a common name for a set of spatial roles but also constrains the space where roles can be enabled. Moreover it specifies the type of logical locations and ultimately the granularity of the position that the users playing that role may occupy. A role instance is a role fulfilling the constraints defined at schema level. A spatial role has thus the same name of the schema role name whereas the spatial boundary of the role is a spatial feature with a precise semantics. It should be noticed that all spatial roles instantiating a role schema are fully identified by the role extent (feature) name. Another important property of the role schema is that it may be assigned permissions. Those permissions are then inherited and shared by all the instances of the role schema.

Users are assigned spatial roles, thus instances of some role schema that can be activated during a session. Unlike RBAC, roles are *enabled* only when the user position is contained in the role extent.

For sake of readability, in what follows we present the model as organized in a number of logical parts, one for each major set of the RBAC model, i.e. roles, permissions, users, sessions. The general structure of the model is illustrated in Figure 1. We use the graphical representation adopted in RBAC. In defining the model, we refer to the notation introduced in the previous section and summarized in Table I.

4.1 Role schemas and instances

A role schema defines a common name for a set of roles, the feature type of the role extent, the feature type of the logical locations and the mapping function relating real positions with logical positions.

Definition 4.1 *Role Schema.* A Role Schema is a tuple $< r, ext, loc, m_{loc} >$ where:

—$r \in R$;

—$ext \in REXT\_FT$;

—$loc \in LPOS\_FT$;

—$loc \subseteq_{ft} ext$;

—$m_{loc} \in M$ is a location mapping function for feature type loc.

We denote with R_S the set of role schemas and we assume that, given a role name $r \in R$, r is unique in R_S. ◇

An example of role schema is the tuple $< Doctor, Hospital, Sector, m_{Sector} >$ in which: $Doctor$ is the name of the role; $Hospital$ the feature type of the role extent, thus the "kind of object" that spatially constrains the role; $Sector$ is the feature type of logical positions, as we suppose that the area of the hospital is subdivided in sectors; finally m_{Sector} the position mapping function that maps a real position into a logical one. Such a function computes the sector containing the real position of the user.

From Definition 4.1 it follows that the feature type representing the logical location must precede in the ordering the feature type representing the role extent. According to Definition 3.3, this means that logical positions must be contained in role extents. Thus, it cannot occur that a location only partially overlaps the space defined by the role extent. From this assumption it follows that it is always possible to determine whether the logical location of a user is contained in a role extent and thus which roles in the session are enabled. Based on the previous definition, the role schema for a role name r is unique. This means that different schemas for the same role, such as: $< Doctor, Hospital, Sector, m_{Sector} >$ and $< Doctor, Department, Room, m_{Room} >$ are not allowed. Should the application require a role on different types of extents, a hierarchy of role schemas has to be defined (see Section 5).

Given a role schema, role instances can be simply created by specifying for the role name its extent as a feature of the type specified in the schema.

Notice that of the four components of a role schema, only the first two are actually needed for the specification of a role instance. As we will see, the last two components, involving the notion of logical position, are needed for role activation. In the following, to indicate the component α of role schema r_s, we use the notation $r_s.\alpha$. A similar notation is used for role instances, introduced by the following definition.

Definition 4.2 *Role Instance*. Given a role schema r_s, an instance r_i of r_s is a pair $< r, e >$ where $r = r_s.r$ and $e \in F$, such that $FT\_Type(e) = r_s.ext$. The schema of r_i is denoted by $SchemaOf(r_i)$.

We denote with $R_I \subseteq R \times REXT$ the set of role instances for all role schemas. For the sake of readability, a role instance $< r, e >$ is also denoted by $r(e)$. ◇

4.2 Permissions

In GEO-RBAC, permissions can be associated either with the role schema and inherited by all role instances of the schema or directly with role instances. Such different granularities are formalized by introducing two functions: $S\_PrmsAssignment$, relating role schemas and permissions sets; $I\_PrmsAssignment$ relating role instances to specific permissions. Function $I\_PrmsAssignment^*$ is then introduced to combine permissions directly assigned to spatial roles with permissions inherited from their role schema.

Definition 4.3 *Permissions*. Let R_S be the set of role schemas, R_I the set of role instances, OPS the set of operations, OBJ the set of objects. The set of permissions $PRMS$ is defined as $PRMS = 2^{(OPS \times OBS)}$. We also define:

—$SPA_S : R_S \times PRMS$, a many-to-many mapping permission-to-spatial role schema assignment relation;

—$S\_PrmsAssignment : R_S \rightarrow 2^{PRMS}$, the mapping of spatial role schemas onto sets of permissions. Given a role schema r_s, $S\_PrmsAssignment(r_s) = \{p \in PRMS | < r_s, p > \in SPA_S\}$;

—$SPA_I : R_I \times PRMS$, a many-to-many mapping permission-to-spatial role instance assignment relation;

—$I\_PrmsAssignment : R_I \rightarrow 2^{PRMS}$ the mapping of spatial role instances onto sets of permissions. Given a role instance r_i, $I\_PrmsAssignment(r_i) = \{p \in PRMS | < r_i, p > \in SPA_I\}$;

—$I\_PrmsAssignment^* : R_I \rightarrow 2^{PRMS}$ such that given a role instance r_i, $I\_PrmsAssignment^*(r_i) = I\_PrmsAssignment(r_i) \cup S\_PrmsAssignment(SchemaOf(r_i))\}$. Hence the permissions of a role are those assigned to its schema plus those directly assigned to the instance. ◇

4.3 Users

Spatial roles are assigned to users. The definition of the model for this part is conceptually analogous to that in RBAC.

Definition 4.4 *Users*. Let U be the set of users and R_I be the set of role instances. We define:

—$SUA \subseteq U \times R_I$, a many-to-many mapping user-to-spatial role instance assignment relation;

—$SR\_AssignedUser : R_I \rightarrow 2^U$, the mapping of spatial role instances onto sets of users. Formally $SR\_AssignedUser(< r, e >) = \{u \in U | (u, < r, e >) \in SUA\}$. ◇

4.4 Sessions

When a user logs in, a new session is activated and a number of roles are selected to be included in the session role set. However, for a session role to be enabled, the user should be logically located within the space of the role extent. In order to compute the logical position of a user playing a role r in a session, the location mapping function defined in the schema of r is applied to the user real position, provided by the external environment. Hence, if the logical position of the user is spatially contained in the extent of r, the role is *enabled*.

Definition 4.5 *Sessions*. Let U be the set of users and SES the set of sessions. We define:

—$SessionUser : SES \rightarrow U$, the mapping from a session s to the user of s;
—$SessionRoles : SES \rightarrow 2^{R_I}$ with $SessionRoles(s) \subseteq \{< r, e > |(SessionUser(s), < r, e >) \in SUA\}$. ◇

$SessionRoles(s)$ corresponds to the roles that can be potentially activated in session s. However, depending on the user position during that session, only a subset of such roles is enabled and permissions granted. To determine enabled roles, containment between logical user position and role extent has to be assessed. Then, for each enabled role, the set of permissions assigned to the corresponding role schema is determined.

Definition 4.6 *Enabled Roles*. Enabled session roles are defined as the function: $EnabledSessionRoles : SES \times RPOS \rightarrow 2^{R_I}$ such that $EnabledSessionRole(s, rp) = \{< r, e > \in R_I | < r, e > \in SessionRoles(s),\ lpos = SchemaOf(< r, e >).m_{loc}(rp), Contains(LocObj(e), LocObj(lpos)) = TRUE\}$. ◇

Enabled roles are the basis for determining whether to grant or reject an access request An access request is a tuple $\langle s, rp, p, o\rangle$ stating that the user of session s located at real position rp wants to perform operation p on object o, thus $\langle s, rp, p, o\rangle \in SES \times RPOS \times OPS \times OBJ$. An access request can be satisfied at real position rp, if permission (p, o) belongs to the set of permissions assigned to the roles that are enabled in s when the session user is in position rp.

Definition 4.7 *Authorization control function*. An access request is a tuple $ar = \langle s, rp, p, o\rangle \in SES \times RPOS \times OPS \times OBJ$. ar can be satisfied at position rp if

$$(p, o) \in \bigcup_{y \in EnabledSessionRoles(s,rp)} I\_PrmsAssignment^*(y) \quad \diamond$$

5. HIERARCHIES IN GEO-RBAC

As Hierarchical RBAC adds to Flat RBAC the support for role hierarchies [Sandhu et al. 2000], Hierarchical GEO-RBAC (GEO-HRAC) adds to GEO-RBAC the support to model hierarchies. According to [Sandhu et al. 2000], the hierarchical level can be defined by introducing a partial order $\preceq$ between roles such that $r_i \preceq r_j$ means that: (i) r_j inherits all permissions assigned to r_i; ii) users which have been assigned r_j have also assigned r_i. In our model, however, the notion of hierarchy which is commonly adopted in RBAC systems is not sufficient to deal with the spatial dimension of roles. Therefore we have defined GEO-HRBAC which extends the concept of hierarchy by introducing two major novelties.

The first novelty concerns the specification of two distinct hierarchies called respectively *role schema hierarchy* and *role instance hierarchy*. We recall that the notion of role schema serves to abstract common properties out of a set of homogeneous roles and thus simplify the specification of roles and ultimately role engineering. The schema hierarchy can be seen as the natural evolution of the concept of schema since it enables the inheritance of permissions and user membership among sets of homogeneous roles and thus further simplifies role definition. On the other side, the role instance hierarchy more closely resembles the conventional notion of hierarchy, in that it defines a partial order over a set of roles. In

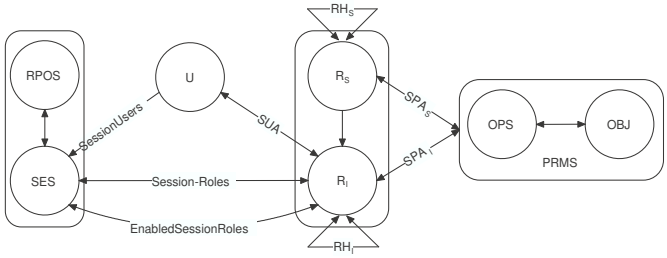

Fig. 2. Hierarchical GEO-RBAC

addition, because of the spatial nature of roles, the hierarchy specifies containment relationships between role extents.

The second extension concerns the modeling of the operations of *role activation* and *enabling*. A role is activated when it is candidate to become enabled; the role is enabled when it becomes effective. The RBAC standard does not include the notion of *enabled* role, and moreover it does not specify how the operation of role activation in Hierarchical RBAC is carried out, i.e. whether the hierarchy is used to determine which roles to activate or not. Therefore, to provide a comprehensive specification of GEO-HRBAC, we present a model in which the role instance hierarchy is used to derive the roles which not only are activated but also enabled in a session. As a result, the specificity of the roles which are effective and, ultimately, the set of permissions which the user is allowed to request may vary transparently with the user's position.

5.1 Schema and role instance hierarchies

The two different hierarchies defined respectively for schemas and role instances as illustrated in Figure 2.

5.1.1 *Role Schema hierarchy.* At the schema level, the hierarchy allows us to define a partial order $\preceq_s$ between role schemas. Such a hierarchy is then inherited at the instance level. Morover, similarly to HRBAC, the partial order is defined according to the semantics of the considered application domain. Given two role schemas r_{s_1} and r_{s_2}, if $r_{s_1} \preceq_s r_{s_2}$ then r_{s_2} inherits all the permissions of r_{s_1}. Furthermore, we assume that the ordering between role schemas can be defined only when a containment relationship holds between the extent and the location types of the role schemas. As an example, consider the schemas:

$$Doc =< Doctor, Hospital, Sector, m_{Sector} >$$

$$Ped =< Pediatrist, Department, Room, m_{Room} >$$

$Doc \preceq_s Ped$ means that pediatrists have at least the same permissions of doctors. Moreover, the feature type $Department$ is contained in $Hospital$, whilst the feature type $Room$ is contained in $Sector$. We assume that the partial order $\preceq_s$ is a bounded lattice. The top and bottom schemas are represented by two system-defined schemas, denoted respectively as *MaxSchema* and *MinSchema*. *MaxSchema* is the schema which precedes all the user-defined schemas in R_S and which is assigned the whole set of permissions. *MinSchema* is preceded by all the user-defined schemas. Further, the set of permissions assigned to

MinSchema is conventionally the empty set. Both the MinSchema and the MaxSchema have unique instances. Further the components of these schemas, like the role extent type, are system-defined (see Section 3). Formally the schema hierarchy is defined as follows.

Definition 5.1 *Schema hierarchy*. The schema hierarchy is defined as the bounded lattice $RH_S = (R_S, \preceq_s)$ such that:

(1) R_S is a set of role schemas, containing the system-defined role schemas $MinSchema \cong < MinS, Top_{ft}, Top_{ft}, m_{top} >$ and $MaxSchema \cong < MaxS, Bot_{ft}, Bot_{ft}, m_{bot} >$, such that:
—$\nexists p \in PRMS$ such that $(MinSchema, p) \in SPA_S$ (i.e., no permission is assigned to $MinSchema$);
—$\forall p \in PRMS$, $(MaxSchema, p) \in SPA_S$ (i.e., all permissions are assigned to $MaxSchema$).

(2) $\preceq_s \subseteq R_S \times R_S$ is a partial order over R_S. If $r_{s_1} \preceq_s r_{s_2}$ holds, $r_{s_2}.ext \subseteq_{ft} r_{s_1}.ext$, and $r_{s_2}.loc \subseteq_{ft} r_{s_1}.loc$ must hold. We assume that:
—$\forall r_s \in R_S,\ MinSchema \preceq_s r_s$;
—$\forall r_s \in R_S,\ r_s \preceq_s MaxSchema$.

(3) $S\_AuthorizedPrms : R_S \rightarrow 2^{PRMS}$ such that, given a role schema r_s, $S\_AuthorizedPrms(r_s)$ returns all permissions assigned to r_s and to all its ancestors, i.e. $S\_AuthorizedPrms(r_s) = \{p \in PRMS | r'_s \preceq_s r_s, < r'_s, p > \in SPA_S\}$. We notice that $S\_AuthorizedPrms(MinSchema) = \emptyset$ and $S\_AuthorizedPrms(MaxSchema) = PRMS$. ◇

5.1.2 *Role instance hierarchy.* In a symmetric way, we assume that a partial order relationship is defined over the set R_I, denoted as $\preceq_i$. For how the model is defined, all the role instances inherit the permissions assigned to their role schema and thus also the permissions of the inherited roles. Given the schemas $Doc \preceq_s Ped$ as previously defined, the role instance $Pediatrist(Dep_1)$ will also inherit the permissions of both the Pediatrist and Doctor role schema. Moreover, if $r_1(g_1)$ and $r_2(g_2)$ are two instances of schemas r_1 and r_2 such that $r_1 \preceq_s r_2$ then $r_1(g_1) \preceq_i r_2(g_2)$ means that not only $r_2(g_2)$ inherits the permissions of the role schema of r_1 but also the permissions that have been assigned specifically to the instance $r_1(g_1)$. Suppose the role instance $Doctor(Hosp_1)$ has been given a specific permission. Then, $Doctor(Hosp_1) \preceq_i Pediatrist(Dep_1)$ means that the pediatrist working at department Dep_1 will also inherit the permissions of the doctors of the $Hosp_1$ hospital in which Dep_1 is located.

Another case to be considered is when the roles are instances of the same role schema and the role extents are related by containment. Given a schema r_s, Consider the instances $r_s(e_1)$ and $r_s(e_2)$ with the geometry of e_2 contained in e_1. Since it seems reasonable that a user playing role $r_s(e_2)$, also plays the role with the larger $r_s(e_1)$, we consider this hierarchy be implicitly defined. Similarly to the role schema hierarchy, we assume that the partial order $\preceq_i$ is a bounded lattice. The top and bottom roles are denoted respectively $MaxIRole$ and $MinIRole$. By convention, these roles are the unique instances of the corresponding $MaxSchema$ and$MinSchema$ in the schema hierarchy. Further, $MaxIRole$ is assigned the whole set of permissions; in contrast, the set of permissions assigned to $MinIRole$ is the empty set. Notice that ***MinIRole*** has as extent the whole reference space, while ***MaxIRole*** has as extent undefined extent. An undefined extent means that no user can be located in it and thus the role can never be enabled. Further, no user is assigned

to *MaxIRole*. Conversely all users are assigned to *MinIRole*. Formally the hierarchy is defined as follows.

Definition 5.2 *Role instance hierarchy*. The role instance hierarchy is defined as the bounded lattice $RH_I = (R_I, \preceq_i)$ such that:

(1) R_I is a set of role instances containing the system-defined role instances $MinIRole \cong < MinS, Top_f >$ and $MaxIRole \cong < MaxS, Bot_f >$, such that:
—$\nexists p \in PRMS$ such that $(MinIRole, p) \in SPA_I$ or $(MaxIRole, p) \in SPA_I$ (i.e., no permission is directly assigned to $MinIRole$ and $MaxIRole$);
—$\forall u \in U, (MinIRole, u) \in SUA$, $\nexists u \in U$ such that $(MaxIRole, u) \in SUA$ (i.e., all users are assigned to $MinIRole$ whereas no user is assigned to $MaxIRole$);

(2) $\preceq_i \subseteq R_I \times R_I$ is a partial order over R_I. The ordering $< r_1, e_1 > \preceq_i < r_2, e_2 >$ holds iff $SchemaOf(< r_1, e_1 >) \preceq_s SchemaOf(< r_2, e_2 >)$ and $LocObj(e_2) \subseteq LocObj(e_1)$. We notice that
—$\forall r_i \in R_I,\ MinIRole \preceq_i r_i$.
—$\forall r_i \in R_I,\ r_i \preceq_i MaxIRole$.

(3) $I\_AuthorizedPrms : R_I \rightarrow 2^{PRMS}$ such that, given a role instance r_i, $I\_AuthorizedPrms(r_i)$ returns all permissions assigned to r_i and to all its ancestors, i.e., $I\_AuthorizedPrms(r_i) = \{p \in PRMS | r'_i \preceq_i r_i, p \in I\_PrmsAssignment^*(r'_i)\}$. We notice that $I\_AuthorizedPrms(MinIRole) = \emptyset$ and $I\_AuthorizedPrms(MaxIRole) = PRMS$.

(4) $I\_AuthorizedUsers : R_I \rightarrow 2^U$ such that, given a role instance r_i, $I\_AuthorizedUsers(r_i)$ returns all users assigned to r_i and to all its descendants, i.e., $I\_AuthorizedUsers(r_i) = \{u \in U | r_i \preceq_i r'_i, < u, r'_i > \in SUA\}$. We notice that $I\_AuthorizedUsers(MinIRole) = U$ and $I\_AuthorizedUsers(MaxIRole) = \emptyset$ ◇

¿From the above definition it follows that the ordering between role schemas corresponds to the ordering of position granularities: the location becomes more precise as the role becomes more specific while the extension gets smaller. That is like to say that the more "powerful" roles are those operating on smaller regions. Moreover, we note that the ordering between instances of the same schema (i.e., spatial roles with the same name) is implicitly defined by the containment relationship existing between their extents.

Finally, we can observe that the assumption that no users are assigned to $MaxIRole$ prevents users from being "automatically" assigned to all roles, as it would happen if a user would be assigned to $MaxIRole$. As we will see later on, this assumption enables the introduction of exclusive role constraints.

Based on the above definition, it is possible to show a number of properties of schema and instance role hierarchies.

PROPOSITION 5.3. *Let $r_{s_1} \in R_S, r_{s_2} \in R_S$, $r_{i_1} \in R_I, r_{i_2} \in R_I$, such that $SchemaOf(r_{i_1}) = r_{s_1}$ and $SchemaOf(r_{i_2}) = r_{s_2}$. Suppose that $r_{s_1} \preceq_s r_{s_2}$ and $r_{i_1} \preceq_i r_{i_2}$. The following properties hold:*

—$S\_AuthorizedPrms(r_{s_1}) \subseteq S\_AuthorizedPrms(r_{s_2})$;
—$I\_AuthorizedPrms(r_{i_1}) \subseteq I\_AuthorizedPrms(r_{i_2})$;
—$I\_AuthorizedUsers(r_{i_2}) \subseteq I\_AuthorizedUsers(r_{i_1})$. ◇

Fig. 3. Role hierarchy (a) and role extents (b).

5.1.3 *Spatial relationships in role instance hierarchy.* According to Definition 5.2, the relationships which hold between roles in the role instance hierarchy are not only of semantic nature, but also of spatial, i.e. geometric type. To point out the nature of these relationships, which are at the basis of the access control mechanism defined in GEO-HRBAC, we discuss the following example.

Consider the set of roles R=$\{A(s_0), B(s_1), C(s_2), D(s_3), E(s_4), F(s_5)\}$ where $X(e)$ denotes the role X with a spatial extent identified by e. We assume that roles are identified by their names. Assume $A(s_0) \preceq B(s_1)$, $A(s_0) \preceq_i C(s_2)$, $A(s_0) \preceq_i F(s_5)$, $B(s_1) \preceq_i D(s_3)$, $B(s_1) \preceq_i E(s_4)$, and $C(s_2) \preceq_i E(s_4)$. We draw the role instance hierarchy without redundant edges through a Hasse diagram. The symbols $\top$ and $\bot$ indicate respectively the MaxIRole and the MinIRole. Conventionally, the bottom role is drawn at the top. The corresponding graph is reported in Figure 3(a).

For sake of readability, in the graphical representation, the nodes of the graph are labeled only with roles names. An arrow from X to Y means $X \preceq_i Y$. We assume that the role extents are spatially represented by the rectangles in Figure 3(b). We can observe that:

—If two roles are comparable, that is the one is the ancestor of the other, also the role extents are comparable with respect to set inclusion. For example, $D \preceq_i B$ implies that the extent of D is contained in the extent of B.

—The containment relationship between extents is not a sufficient condition for the roles to be comparable, since the role ordering is application-dependent. In the example, the extent of role F is contained in that of C, but the roles are not comparable.

—The extents of two non-comparable roles, such as roles B and C, can overlap. Therefore, if a user is located in the intersection area of two roles, both of them will be enabled.

5.2 Access Control Function

We now address issues related to the activation and enabling of roles in the case of a role instance hierarchy (simply *hierarchy* hereinafter). The ultimate goal is to define the access control function for GEO-HRBAC. We recall that a role r is *activated* in a session s when r belongs to the set of roles selected in s. The function $SessionRoles(s)$ returns the set of activated roles. The role is *enabled* in position rp when rp falls inside the spatial extent of r. The function $EnabledSessionRoles(s, rp)$ returns the subset of activated roles which are enabled in rp.

In order to define the access control function for GEO-HRBAC, it seems reasonable to assume that if a role r is activated, then all the ancestors of r in the hierarchy are

activated in the same session. For example, consider the ordering $Doctor(Hosp_1) \preceq_i Pediatrist(Dep_1)$. If $Pediatrist(Dep_1)$ is activated then $Doctor(Hosp_1)$ is activated as well. Therefore, roles are automatically activated based on the activation of other child roles. To formalize the access control function for GEO-HRBAC, we introduce two additional functions. The first function, called $SessionRoles^+$ returns the set of roles which are activated either directly by the user or indirectly through the activation of other roles. The second function $EnabledSessionRoles^+$ returns the set of roles, among those activated, that are enabled in rp.

Definition 5.4. We define:

(1) $SessionRoles^+ : SES \rightarrow 2^{R_I}$ such that $SessionRoles^+(s) = \{< r, e >\in R_I| < r, e > \preceq_i < r', e' >, < r', e' >\in SessionRoles(s))\}$.
(2) $EnabledSessionRoles^+ : SES \times RPOS \rightarrow 2^{R_I}$ such that $EnabledSessionRole^+(s, rp) = \{< r, e >\in R_I| < r, e >\in SessionRoles^+(s), lpos = SchemaOf(< r, e >).m_{loc}(rp), Contains(LocObj(e), LocObj(lpos)) = TRUE\}$. ◇

Based on the above definition, we can show that if a role is enabled in a given position, also the ancestor roles in the hierarchy are enabled. In the previous example, it means that if $Pediatrist(Dep_1)$ is enabled in a certain position, then also $Doctor(Hosp_1)$ is enabled. The property is formally defined as follows (see the Appendix for the proof).

PROPOSITION 5.5. *The following properties hold:*

(*1*) $EnabledSessionRoles^+(s, rp) \subseteq SessionRoles^+(s)$.
(*2*) *Let* $r_{i_1}, r_{i_2} \in R_I$ *and suppose that* $r_{i_1} \preceq_i r_{i_2}$. *Then* $r_{i_2} \in EnabledSessionRole^+(s, rp)$ *implies* $r_{i_1} \in EnabledSessionRole^+(s, rp)$.
(*3*) $MaxIRole \notin EnabledSessionRoles(s, rp)$ *for all sessions and positions.* ◇

Based on the previous definitions and properties, we can now define the authorization control function in GEO-HRBAC. The function is similar to the one defined at Core level in that it matches the requested permission against the set of permissions assigned to the roles which are enabled in a certain position. If the matching is successful the permission is granted otherwise the access request is denied. The authorization control function in GEO-HRBAC differs from the one defined in the Core model because the set of roles which are enabled, i.e. $EnabledSessionRoles^+(s, rp)$ consists of session roles and their ancestors.

Definition 5.6 *Authorization control function in GEO-HRBAC.* Consider an access request $ar = \langle s, rp, p, o\rangle \in SES \times RPOS \times OPS \times OBJ$. ar can be satisfied at position rp if

$$(p, o) \in \bigcup_{y \in EnabledSessionRoles^+(s,rp)} I\_PrmsAssignment^*(y). \qquad \diamond$$

We can observe that the roles which are enabled change dynamically, and also that the specificity or granularity of these roles varies with the user position. In other words, the set of permissions that the user selects through the session roles are made differently available based on the user's position.

From Definition 5.6, it follows that, in order to compute the authorization control function, all roles $y \in EnabledSessionRoles^+(s, rp)$ have to be computed. However, depending on the existing hierarchies, such set of roles could be very large and therefore the

Fig. 4. Role hierarchy (a) user's positions in space (b).

computation could be very expensive. In order to reduce this cost, in the following we show that, in order to compute the access control function, the requested permission has not to be matched against every role of the set $EnabledSessionRole^{+}(s, rp)$. Rather, only roles which do not have descendants which are enabled have to be taken into account. We call these roles *the most specific roles*. This suggests a more efficient algorithm for the computation of the access control function. The function MSR which computes the set of most specific roles is defined as $MRS : SES \times RPOS \rightarrow 2^{R_I}$ such that $MSR(s, rp) = \{r \in EnabledSessionRoles^{+}(s, rp) | \nexists r' \in EnabledSessionRoles^{+}(s, rp), r \preceq_i r'\}$. The following example shows intuitively the meaning of MRS.

Consider the role instance graph presented in Section 5.1.3 and assume a session s with $SessionRoles(s) = \{D, E\}$. Assume also that the user is initially in position *p* then moves to *q* and finally to *r* (Figure 4). Based on Definition 5.4, $SessionRoles^{+}(s) = \{\bot, A, B, C, D, E\}$. We now specify the set of enabled roles and the set of most specific roles in each position. Assume the user in position p. Then:

$$EnabledSessionRoles^{+}(s, p) = \{\bot, A, B, D\} \qquad MSR(s, p) = \{D\}$$

If the user moves from p to q, then :

$$EnabledSessionRoles^{+}(s, q) = \{\bot, A, B\} \qquad MSR(s, q) = \{B\}$$

Finally the user moves from q to r. Then:

$$EnabledSessionRoles^{+}(s, r) = \{\bot, C, A\} \qquad MSR(s, r) = \{C\}$$

We show now that in order to compute the access control function, it is sufficient to determine the set of most specific roles and then match the requested permission against the permissions assigned to each of them. Formally this property is specified as follow (see the Appendix for the proof).

THEOREM 5.7. *Consider an access request* $ar = \langle s, rp, p, o\rangle \in SES \times RPOS \times OPS \times OBJ$. *Denote with* $CheckMRS$ *and CheckAll the following predicates:*

—$CheckMRS(p, o) = TRUE$ *iff* $(p, o) \in \bigcup_{y \in MRS(s,rp)} I\_PrmsAssignment^{*}(y)$

—$CheckAll(p, o) = TRUE$ *iff* $(p, o) \in \bigcup_{y \in EnabledSessionRoles^{+}(s,rp)} I\_PrmsAssignment^{*}(y)$

It holds: $CheckMRS(p, o)$ *iff* $CheckAll(p, o)$. ◇

From the above proposition, it follows that the most specific roles are the roles which do not contain any other extent of enabled roles.

Basic objects

$FT = \{Hospital, Dept, Room, Sector, PatientRecord, Map, Person)\}$ with:

$Dept \subseteq_{ft} Hospital, Room \subseteq_{ft} Sector, Room \subseteq_{ft} Dept, Sector \subseteq_{ft} Hospital$

$OBJ = \{Ext(PatientRecord), Ext(Map), Ext(Person)\}$

$OPS = \{GetPatientRecord, UpdatePatientRecord, FindPersonnel, GetMap, GetStatistics\}$

$$PRMS = \{p_1, p_2, p_3, p_4, p_5\} \text{ with } \begin{cases} p_1 = (GetPatientRecord, Ext(PatientRecord)) \\ p_2 = (UpdatePatientRecord, Ext(PatientRecord)) \\ p_3 = (GetMap, Ext(Map)) \\ p_4 = (GetStatistics, Ext(PatientRecord)) \\ p_5 = (FindPersonnel, Ext(Person)) \end{cases}$$

Schema

$R = \{Personnel, Manager, Doctor, Pediatrist, Nurse, Patient\}$

$REXT\_FT = \{Hospital, Dept\}$

$LPOS\_FT = \{Room, Sector\}$

$$R_S = \{Pe, Do, Pd, Nu, Ma, Pe\} \text{ with } \begin{cases} Pe =< Personnel, Hospital, Sector, m_{Sector} > \\ Ma =< Manager, Hospital, Sector, m_{Sector} > \\ Do =< Doctor, Hospital, Sector, m_{Sector} > \\ Pd =< Pediatrist, Dept, Room, m_{Room} > \\ Nu =< Nurse, Dept, Room, m_{Room} > \\ Pa =< Patient, Hospital, Sector, m_{Sector} > \end{cases}$$

Instances

$REXT = \{Hosp_1, Dep_1\}$

$$R_I = \{r_{Pe}, r_{Ma}, r_{Do}, r_{Pd}, r_{Nu}, r_{Pa}\} \text{ with } \begin{cases} r_{Pe} = Personnel(Hosp_1) \\ r_{Ma} = Manager(Hosp_1) \\ r_{Do} = Doctor(Hosp_1) \\ r_{Pd} = Pediatrist(Dep_1) \\ r_{Nu} = Nurse(Dep_1) \\ r_{Pa} = Patient(Hosp_1) \end{cases}$$

Schema role hierarchy

$Pe \preceq_s Ma; Pe \preceq_s Nu; Pe \preceq_s Do \preceq_s Pd$

Instance role hierarchy

$r_{Pe} \preceq_i r_{Ma}; r_{Pe} \preceq_i r_{Nu}; r_{Pe} \preceq_i r_{Do} \preceq_i r_{Pd}$

Permission assignment

$SPA_S = \{(Pe, p_5), (Ma, p_4), (Do, p_1), (Pd, p_2), (Nu, p_1), (Pa, p_3)\}$

User assignment

$U = \{Alice, Sara\}$

$$SUA = \{s_{ua_1}, s_{ua_2}\} \text{ with } \begin{cases} s_{ua_1} = \langle Alice, Pediatrist(Dep_1)\rangle \\ s_{ua_2} = \langle Sara, Nurse(Dep_1)\rangle \end{cases}$$

Sessions

$SES = \{s_1\}, UserSession(s_1) = \{Alice\}$

$SessionRoles(s_1) = \{Pediatrist(Dep_1)\}$

$SessionRoles^+(s_1) = \{Personnel(Hosp_1), Doctor(Hosp_1), Pediatrist(Dep_1)\}$

EnabledRoles

$EnabledSessionRoles(s_1, loc_1) = \{Pediatrist(Dep_1)\}$ if Alice is in Dep_1

$EnabledSessionRoles^+(s_1, loc_1) = \{Personnel(Hosp_1), Doctor(Hosp_1), Pediatrist(Dep_1)\}$

Fig. 5. An example of a GEO-HRBAC application

6. A COMPREHENSIVE EXAMPLE

Before proceeding with the introduction of the third level of our model,i.e. the Constrained GEO-RBAC, we discuss an extended example of access control policy based on GEO-

HBAC (Figure 5). The purpose of the example is twofold: to summarize the concepts of the model introduced so far; and introduce a scenario for the examples that will follow.

We assume the general context introduced in Section 1, concerning a mobile application for the personnel and patients of an hospital. Consider the feature types: $\{Hospital, Dept, Room, Sector, PatientRecord, Map, Person\}$ and assume that the considered objects correspond to the extensions of feature types $PatientRecord$, Map and $Person$. Assume, moreover, that permissions are defined to retrieve and modify records of patients, to compute statistics on patients, to have map-based guidance in the hospital and to find where personnel is located in the hospital.

In this scenario, we consider the roles: *Personnel, Manager, Doctor, Pediatrist, Nurse* and *Patient*. The role schema of, say, $Pediatrist$ is defined as $< Pediatrist, Dept, Room, m_{Room} >$. The schema hierarchy specifies that the schemas of roles *Doctor*, *Nurse* and *Manager* are all preceded by *Personnel*. Moreover *Doctor* precedes *Pediatrist*. A unique permission is assigned to schema $Personnel$. This permission is thus inherited by: $Nurse$ and $Doctor$ which in addition are also authorized to get records of patients; $Pediatrist$ which is allowed to modify such records for the patients of a department; $Manager$ which is allowed to request statistics about patients.

Assumed the role extents $Hosp_1$, which identifies a specific hospital, and Dep_1 a department of $Hosp_1$, we define the role instances: $Personnel(Hosp_1), Manager(Hosp_1), Doctor(Hosp_1), Pediatrist(Dep_1), Nurse(Dep_1), Patient(Hosp_1)$. These roles inherit the permissions assigned to the respective schemas. The role instance hierarchy specifies the ordering of roles, in compliance with the schema hierarchy. In such a hierarchy, the role $Pediatrist(Dep_1)$ is preceded in the ordering by $Doctor(Hosp_1)$ and *Personnel*$(Hosp_1)$.

Now consider users, say $Alice$ and $Sara$ where $Alice$ is a pediatrist in Dep_1 whereas $Sara$ is a nurse in the same department. Now suppose that $Alice$ starts a session s_1 in (real) position loc. We can observe that if Alice is located in the department Dep_1, she is authorized to get and modify the record of the patients that are hosted in that department; conversely, if Alice is somewhere else in the hospital she is not allowed to update such records.

7. CONSTRAINED GEO-RBAC

We now introduce the third component of the GEO-RBAC model called *Constrained GEO-RBAC*. Separation of Duty (SoD) is widely recognized to be a fundamental principle in computer security [Li et al. 2004]. Accordingly, Constrained GEO-RBAC is defined to allow the specification of separation of duty constraints among spatial roles and role schemas. Commonly, Separation of Duty (SoD) constraints are introduced to prevent conflicts of interest arising when a single individual can simultaneously perform sensitive tasks requiring the use of mutually exclusive duties. In the RBAC standard, enforcement of SoD policies is realized by specifying exclusive role constraints. The general form of a role exclusive constraint is: $(\{r_1, ..., r_m\}, n)$ where each r_i is a role and n and m are integers with $n \leq m$. This constraints forbids a user to be a member of n or more roles in $\{r_1, ..., r_m\}$ [Li et al. 2004]. SoD may be defined as Static (SSoD) and Dynamic (DSoD) depending on whether exclusive role constraints are evaluated against the user-role assignment set or against the set of roles activated in user's session [Ferraiolo et al. 2001].

The question that naturally raises is whether in a mobile context, like the one we have

assumed, in which the user's roles are dependent on the position, the notion of SoD is still meaningful. Whether the contextual dimension makes sense for the concept of conflict of interest is an issue of ontological relevance that has not been addressed yet in sufficient depth. On the other hand, there is evidence that exclusive role constraints are important to support the definition and maintenance of access control policy even in the mobile context. This empirical observation has led us to define exclusive role constraints for spatial roles. We denote this class of constraints as Mutually Exclusive Spatial Roles (MESR) constraints. Two or more spatial roles which are mutually exclusive are also said to be *incompatible*.

Constrained GEO-RBAC enables the specification of a rich set of MESR constraints (simply constraints, hereinafter) over schemas and role instances. Among these, *spatial constraints* state that roles are incompatible when played in certain locations. We describe intuitively this concept through an example: consider two roles, *Doctor* and *Manager* defined over a common extent representing a hospital. It seems reasonable to assume that an individual should not be authorized to play the role of Manager in two different locations, i.e. hospitals; similarly, one should not be authorized to play the roles of Doctor and Manager in the same location. In the latter case, we observe that roles become incompatible when the extents of *Doctor* and *Manager* satisfy a precise topological relationship, specifically their extents coincide. On the other hand, there are also cases in which conflicts arise because roles are contemporaneously enabled. For example, a doctor in a hospital cannot be at the same time a patient of such hospital though the individual has been assigned both roles.

For a systematic description of MESR constraints, in what follows we preliminarily introduce a classification of constraints and then we present each class in more detail. The examples that will be shown assume the scenario described in Figure 5. We then show how constraints can be specified in presence of hierarchies. Then, in the next section properties of Constrained GEO-RBAC will be presented.

7.1 The constraint classification

We propose a classification based on three orthogonal criteria which define on one side the *granularity* of the constraint, on the other the *dimension*, spatial and non-spatial, of the constraint and finally when the constraints are verified, i.e *verification time*.

Granularity. Constraints can be specified for both role instances and role schemas. When defined at instance level, the roles to which the constraint apply are to be explicitly enumerated. For example the constraint: $(\{r_1, r_2\}, 2)$ means that the two roles r_1, r_2 are mutually exclusive. Conversely, when an exclusive role constraint is specified between schemas, it means that the constraint is implicitly applied to instances of the schemas' extensions. In particular, given two schemas, rs_1, rs_2, the constraint $(\{rs_1, rs_2\}, 2)$ means that all pair of roles (r_1, r_2) referring respectively to schema rs_1 and rs_2 are mutually exclusive. Note that the constraint applied to a unique schema $(\{rs_1\}, 2)$ means that a user cannot play more that one role instance of schema rs_1. The constraints at schema level enable thus a more compact representation of generalized constraints over sets of roles, that otherwise would require a lengthy specification, simplifying thus role engineering and administration.

Dimension. Constraints may have a spatial dimension or not. Non-spatial constraints are those which are close to the standard RBAC constraints since they do not consider the

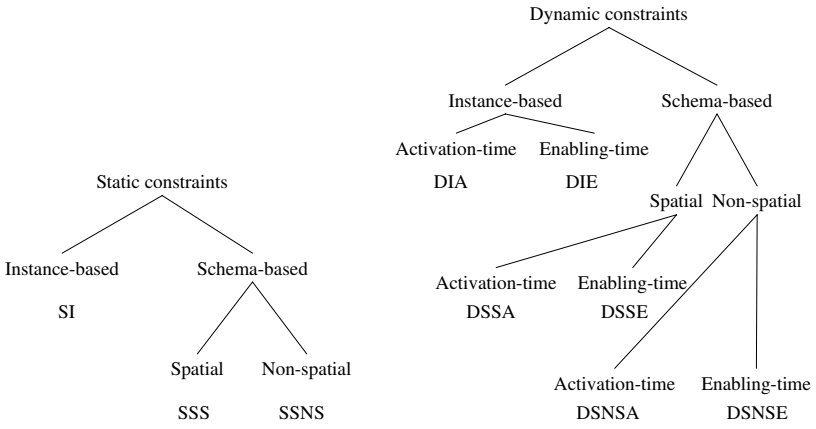

Fig. 6. A graphical representation of constraint classes in GEO-RBAC

spatial dimension of roles. A constraint is spatial when it is applied to the role instances which fulfill a given spatial relationships. In the context of spatial constraints, the intuition is that conflicts of interest may arise not only because of the semantics of roles but also because roles are defined over extents satisfying a given spatial condition. It is the case, for example, of roles which cannot be played over the same region. A typical spatial constraint has the form (rs_1, rs_2, rel) such that rs_1 and rs_2 are role schemas and $rel \in REL$ is the spatial relationships according to which the constraint is evaluated. This constraint states that all pairs of role instances from the specified schemas satisfying relationship rel are mutually exclusive. The spatial relationships we consider are the topological ones introduced in Section 3.

Verification time. Constraints may be evaluated statically or at run time. In the latter case we distinguish two cases, depending on whether the constraint holds at activation-time or enabling-time. We recall the a role, though activated, does not become really effective until it is enabled and that the enabling of a role occurs when the user is located in the role's extent. Dynamic constraints at activation time, like conventional dynamic SoD, prevent the user from selecting incompatible roles in the same session. Dynamic constraints at enabling time introduce an additional and more detailed level of specification, in the sense that prevent the user from playing contemporaneously two activated roles.

By combining the previous three criteria, we obtain several classes of constraints, graphically represented in Figure 6. Each node of the tree represents a class of constraints. The children of a given node in the tree represent a partition in sub-classes of the class of constraints represented by the father node. The resulting partition is characterized by the classes corresponding to the leaves. For each leaf, the figure presents the notation we use in the following to identify the corresponding class of constraints. Each class of constraints will be formally defined in the following. In order to distinguish among static or dynamic constraints, at activation or at enabling time, we adopt the following notation. With $(set, n)_{type}$, $type \in \{\perp, a, e\}$ we denote:

(1) instance-based constraints if set is a set of role instances;

(2) schema-based constraints if set is a set of role schemas;

Class	Form	Formal specification	E
SI	$(RoleSet, n)_{\perp}$	$\forall h \subseteq RoleSet, \mid h \mid \geq n \Rightarrow \bigcap_{r \in h} SR\_AssignedUser(r) = \emptyset$	1
SSNS	$(SchemaSet, n)_{\perp}$ $\mid SchemaSet \mid > 1$	$\left.\begin{array}{l} \forall s = \{rs_i, \ldots, rs_j\}, \\ s \subseteq SchemaSet, \mid s \mid \geq n \end{array}\right\} \Rightarrow \bigcap_{i \leq k \leq j} \left(\bigcup_{r \in Ext(rs_k)} SR\_AssignedUser(r) \right) = \emptyset$	2
	$(\{rs\}, n)_{\perp}$	$\forall h \subseteq Ext(rs), \mid h \mid \geq n \Rightarrow \bigcap_{r \in h} SR\_AssignedUser(r) = \emptyset$	3
SSS	$(rs_1, rs_2, rel)_{\perp}$	$\left.\begin{array}{l} \forall x \in Ext(rs_1), \forall y \in Ext(rs_2), \\ x \ rel \ y \end{array}\right\} \Rightarrow SR\_AssignedUser(x) \bigcap SR\_AssignedUser(y) = \emptyset$	4

Table II. Static constraints formal specification

(3) static constraints if $type = \perp$;

(4) dynamic constraints at activation (enabling) time if $type = a$ ($type = e$).

We extend in analogous way this notation to the case of spatial constraints.

We observe that spatial instance-based constraints are not considered since their specification should result in trivial constraints. In fact, differently from schema-based constraints, at instance level, we exactly know which are the extents of the given role instances, and thus we can always determine if the used spatial relationship is satisfied or not.

7.2 Static constraints

In the following, we present the classes of constraints expressing classical static separation of duty. The logical formulas corresponding to each constraints are presented in Table II.

The first class of constraints we consider is that of *constraints defined at instance level* (*SI constraints*). SI constraints are those which more closely resemble the classical RBAC static separation of duty constraints. The purpose of a SI constraint $(role\_instance\_set, n)_{\perp}$ is to prevent the user from playing *n* or more role instances among those specified in the $role\_instance\_set$.

Definition 7.1 *SI constraints.* Let $RoleSet = \{r_1(e_1), \ldots, r_k(e_k)\} \subseteq R_I$ be a set of role instances. Then, $c \equiv (RoleSet, n)_{\perp}$, $2 \leq n \leq k$, is a constraint in SI, specifying that, for any subset h of $RoleSet$ having cardinality at least n, no user is assigned to all roles in the subset (E1 in Table II). ◇

EXAMPLE 1. *Consider the set* $RoleSet = \{Doctor(Hosp_1), Doctor(Hosp_2)\}$. *According to Definition 7.1, the constraint* $(RoleSet, 2)_{\perp} \in SI$ *means that an individual cannot be doctor in both the specified hospitals* $Hosp_1$ *and* $Hosp_2$. *Note that roles can be instances of either the same or different schemas.* ◇

The next class of static constraints consists of *non-spatial constraints defined at schema level* (*SSNS constraints*). Basically, a SSNS constraint states that an individual cannot play a certain number of roles from some given schemas. More precisely, the purpose of a SSNS constraint $(role\_schema\_set, n)_{\perp}$, $n \geq 2$, is to prevent the user from playing *n* distinct role instances from n schemas in $role\_schema\_set$. When $role\_schema\_set$ contains just one role schema, say rs, the meaning is that a user cannot play n with $n \geq 2$ role instances of rs. Thus, what changes in the two cases is the composition of the roles set to be controlled: in the first case it is composed of role instances having all different schemas whereas in the other case all instances have the same schema. The last case is

interesting since it seems reasonable in practice to constrain an individual to play a given role in a unique or however limited number of spatial extents.

Definition 7.2 *SSNS constraints*. Let $SchemaSet = \{rs_1, \ldots, rs_k\} \subseteq R_S$ be a set of role schemas. Then, $c \equiv (SchemaSet, n)_\perp$, $2 \leq n \leq k$, is a constraint in $SSNS$ having the following meaning:

—if $\mid SchemaSet \mid \geq 2$, for any subset of $SchemaSet$ having cardinality at least n, no user is assigned to at least one role for each schema in the subset (E2 in Table II).

—if $\mid SchemaSet \mid = 1$, i.e., $SchemaSet = \{rs\}$, no user has been assigned to at least n role instances of rs (E3 in Table II). ◇

EXAMPLE 2. *Consider the role schema* $Do =< Doctor, Hospital, Sector, m_{Sector} >$. *The constraint* $(Do, 2)_\perp \in SSNS$ *means that an individual can be doctor in at most one hospital.* ◇

The basic idea of *schema-based spatial constraints* (*SSS constraints*) is to prevent a user from playing some role instances if the role extents fulfill some specified spatial relationship. Such constraints could be useful since, because roles are defined over spatial extents, it may be the case that conflicts of interest arise when roles are played over spatial interrelated regions. For example, an individual cannot be doctor and manager in the same hospital. For the sake of simplicity, we assume that a SSS constraint is defined upon two role schemas and the spatial relation is one of the topological relationship in REL, introduced in Section 3. Formally, a SSS constraint can be defined as follows.

Definition 7.3 *SSS constraints*. Let $rs_1 \in R_S$ and $rs_2 \in R_S$ be two role schemas and $rel \in REL$. Then, $c \equiv (rs_1, rs_2, rel)_\perp$ is a constraint in SSS specifying that no user is assigned to role instances r_1 and r_2, such that r_1 is an instance of rs_1, r_2 is an instance of rs_2 and the extents of r_1 and r_2 satisfy the spatial relationship rel (E1 in Table II). ◇

EXAMPLE 3. *Consider the schemas denoted with* Do *and* Ma *corresponding to roles* Doctor *and* Manager. *Then the constraint* $(Do, Ma, Equal)_\perp$ *means that an individual cannot be doctor and manager in the same hospital. Note that* $Equal \in REL$ *is a topological relationship.* ◇

7.3 Dynamic constraints

In that follows, we present the dynamic constraint classes used to express classical dynamic separation of duty in GEO-RBAC. The logical formulas corresponding to each constraints are presented in Table III. In presenting constraints, we distinguish between constraints verified at activation time and those verified at enabling time.

The first couple of dynamic constraint classes we consider is that consisting of *constraints defined at instance level* and verified at activation time (DIA constraints) or at enabling time (DIE constraints). The meaning of a generic dynamic constraint at instance level $(role\_instance\_set, n)_f$, $f \in \{a, e\}$, is to prevent a user from activating (case $f = a$) or enabling (case $f = e$) n or more role instances among those specified in $role\_instance\_set$. DIA constraints are those which more closely resemble the classical RBAC dynamic separation of duty constraints.

Definition 7.4 *DIA-DIE constraints*. Let $RoleSet = \{r_1(e_1), \ldots, r_k(e_k)\} \subseteq R_I$ be a set of role instances. Then, $c \equiv (RoleSet, n)_f$, $2 \leq n \leq k$, $f \in \{a, e\}$ is a dynamic instance-based constraint having the following meaning:

Class	Form	Formal specification	E
DIA	$(RoleSet, n)_a$	$\forall t \in SES, \forall h \subseteq RoleSet, h \subseteq SessionRoles(t) \Rightarrow \mid h \mid < n$	1
DIE	$(RoleSet, n)_e$	$\left.\begin{array}{l}\forall t \in SES, \forall h \subseteq RoleSet, \forall pos \in RPOS, \\ h \subseteq EnabledSessionRoles(t, pos)\end{array}\right\} \Rightarrow \mid h \mid < n$	2
DSNSA	$(SchemaSet, n)_a$ $\mid SchemaSet \mid > 1$	$\left.\begin{array}{l}\forall t \in SES, \forall s = \{rs_i, \ldots, rs_j\}, s \subseteq SchemaSet, \\ \forall h = \{r_i(e_i), \ldots, r_j(e_j)\}, r_k \in Ext(rs_k), i \leq j \leq k, \\ h \subseteq SessionRoles(t)\end{array}\right\} \Rightarrow \mid h \mid < n$	3
	$(\{rs\}, n)_a$	$\forall t \in SES, \forall h \subseteq Ext(rs), h \subseteq SessionRoles(t) \Rightarrow \mid h \mid < n$	4
DSNSE	$(SchemaSet, n)_e$ $\mid SchemaSet \mid > 1$	$\left.\begin{array}{l}\forall t \in SES, \forall s = \{rs_i, \ldots, rs_j\}, s \subseteq SchemaSet, \forall pos \in RPOS, \forall h = \{r_i(e_i), \ldots, r_j(e_j)\}, \\ r_k \in Ext(rs_k), i \leq j \leq k, h \subseteq EnabledSessionRoles(t, pos)\end{array}\right\} \Rightarrow \mid h \mid < n$	5
	$(\{rs\}, n)_e$	$\left.\begin{array}{l}\forall t \in SES, \forall pos \in RPOS, \forall h \subseteq Ext(rs), \\ h \subseteq EnabledSessionRoles(t, rpos)\end{array}\right\} \Rightarrow \mid h \mid < n$	6
DSSA	$(rs_1, rs_2, rel)_a$	$\left.\begin{array}{l}\forall x \in Ext(rs_1), \forall y \in Ext(rs_2), \\ \forall t \in SES, x \ Rel \ y\end{array}\right\} \Rightarrow \{x, y\} \not\subset SessionRoles(t)$	7
DSSE	$(rs_1, rs_2, rel)_e$	$\left.\begin{array}{l}\forall x \in Ext(rs_1), \forall y \in Ext(rs_2), \\ \forall t \in SES, \forall pos \in RPOS, x \ Rel \ y\end{array}\right\} \Rightarrow \{x, y\} \not\subset EnabledSessionRoles(t, pos)$	8

Table III. Dynamic constraints formal specification

—$(RoleSet, n)_a \in DIA$ means that for any subset h of $RoleSet$ having a cardinality at least n, there is no session in which at least n roles from h have been activated (E1 in Table III).

—$(RoleSet, n)_e \in DIE$ means that for any subset h of $RoleSet$ having a cardinality at least n, there is no session in which at least n roles from h have been enabled (E2 in Table III). ◇

EXAMPLE 4. *Consider the following set of instances of the $Nurse$ role: $RoleSet = \{Nurse(Dep_1), Nurse(Dep_2)\}$. The constraint $(RoleSet, 2)_a$ means that an individual cannot activate both roles in the session, in other words that a nurse cannot be active in both departments Dep_1 and Dep_2 during the working time.* ◇

The second couple of dynamic constraint classes we consider is that consisting of *non-spatial constraints defined at schema level* and verified at activation time ($DSNSA$ constraints) or at enabling time ($DSNSE$ constraints). The meaning of a generic dynamic non-spatial constraint at schema level $(role\_schema\_set, n)_f$, $n \geq 2$, $f \in \{a, e\}$, is to prevent a user from activating (case $f = a$) or enabling (case $f = e$) within a session n distinct role instances from n schemas in $role\_schema\_set$. Analogously to what happens in the static case, when $role\_schema\_set$ is composed by exactly one schema, the meaning of the constraint is to prevent the activation or the enabling, depending on which case we are considering, of n roles instances of the given schema within a session.

Definition 7.5 *DSNSA-DSNSE constraints*. Let $SchemaSet = \{rs_1, \ldots, rs_k\} \subseteq R_S$ be a set of role schemas. Then, $c \equiv (SchemaSet, n)_f$, $2 \leq n \leq k$, $f \in \{a, e\}$, is a dynamic non-spatial schema-based constraint having the following meaning:

—$(SchemaSet, n)_a \in DSNSA$ with $\mid SchemaSet \mid \geq 2$ means that for any subset of $SchemaSet$ having cardinality at least n, no user has activated at least one role for each schema in the subset within a session (E3 in Table III).

—$(SchemaSet, n)_a \in DSNSA$ with $\mid SchemaSet \mid = 1$, i.e. $SchemaSet = \{rs\}$ means that no user has activated at least n instances of rs within a session (E4 in Table III).

—$(SchemaSet, n)_e \in DSNSE$ with $\mid SchemaSet \mid \geq 2$ means that for any subset of $SchemaSet$ having cardinality at least n, no user has enabled at least one role for each schema in the subset within a session (E5 in Table III).

—$(SchemaSet, n)_e \in DSNSE$ with $\mid SchemaSet \mid = 1$, i.e. $SchemaSet = \{rs\}$ means that no user has activated at least n instances of rs within a session (E6 in Table III). ◇

EXAMPLE 5. *Consider the schema defined for the* nurse *role, denoted by Nu. The constraint $(\{Nu\}, 2)_a$ states that a nurse can still be assigned to different departments but cannot be active in more than one department during the session of interaction with the system. The constraint at enabling time $(\{Nu\}, 2)_e$ states that a nurse can be active in more than one department and thus play different roles of the same schema, but should the departments share a common space and the individual be located there, only one of such roles can be enabled.* ◇

Finally, last couple of dynamic constraint classes we consider is that consisting of *spatial constraints defined at schema level* and verified at activation time ($DSNSA$ constraints) or at enabling time ($DSNSE$ constraints). The meaning of a generic dynamic spatial constraint at schema level $(rs_1, rs_2, rel)_f$, $f \in \{a, e\}$, is to prevent a user from activating (case $f = a$) or enabling (case $f = e$) within a session distinct role instances from rs_1 and rs_2 schemas when their extents satisfy the spatial relationship rel.

Definition 7.6 *DSSA-DSSE constraints*. Let rs_1 and rs_2 be two role schemas and $rel \in REL$. Then, $c \equiv (rs_1, rs_2, rel)_f$, $f \in \{a, e\}$, is a dynamic spatial schema-based constraint having the following meaning:

—$(rs_1, rs_2, rel)_a \in DSSA$ means that no user has activated within a session both roles instances r_1 and r_2, such that r_1 is an instance of rs_1, r_2 is an instance of rs_2 and the extents of r_1 and r_2 satisfy the spatial relationship rel (E7 in Table III).

—$(rs_1, rs_2, rel)_e \in DSSE$ means that no two roles r_1 and r_2, with r_1 instance of rs_1 and r_2 instance of rs_2, can be both enabled in the same session, if the extents of r_1 and r_2 satisfy the spatial relationship rel (E8 in Table III).

EXAMPLE 6. *Consider the schemas Do and Pa corresponding to roles $Doctor$ and $Patient$. Then the constraint is: $(Do, Pa, Equal)_a$ means that a doctor cannot be simultaneously a patient in the same hospital.* ◇

7.4 Constraints for GEO-HRBAC

When considering GEO-RBAC, all the constraints presented in the previous sections have to be revised, in order to take into account inherited information, in terms of authorized users and activated sessions, during constraint specification and checking. More precisely, GEO-RBAC constraints have to be modified as follows:

—*Static constraints*. Static constraints, in order to prevent a user from playing mutually exclusive role instances, consider the set of roles to which a user has been directly assigned because of the use of the $SR\_AssignedUser$ function. In presence of role hierarchies, due to the inheritance from descendant to ancestor roles of the authorized users, in the equations in Table II function $SR\_AssignedUser$ has to be replaced by function $I\_AuthorizedUser$.

—*Dynamic constraints*. Dynamic constraints, in order to prevent a user from playing mutually exclusive role instances inside a session, consider the set of activated or enabled roles selected among those to which a user has been directly assigned because of the use of functions $SessionRoles$ and $EnabledSessionRoles$. In presence of role hierarchies, since ancestor roles are activated (enabled) when descendant roles are activated (enabled), in the equations in Table III functions $SessionRoles$ and $EnabledSessionRoles$ have to be replaced by functions $SessionRoles^+$ and $EnabledSessionRoles^+$.

EXAMPLE 7. *Consider the static constraint* $c_1 \equiv (RoleSet, 2)_\perp$, $RoleSet = \{Doctor(Hosp_1), Doctor(Hosp_2)\}$, *of Example 1. Without hierarchies, the meaning of this constraint is that of preventing an individual to be doctor in both the specified hospitals* $Hosp_1$ *and* $Hosp_2$. *In presence of the role-schema hierarchy pointed out in Figure 5, since* $< Personnel, Hospital, Sector, m_{Sector} > \preceq_s < Doctor, Hospital, Sector, m_{Sector} > \preceq_s < Pediatrist, Dept, Room, m_{Room} >$, *the meaning of this constraint becomes that of preventing an individual to be doctor in hospital* $Hosp_1$ *or pediatrist in a department of hospital* $Hosp_1$ *and at the same time be doctor in hospital* $Hosp_2$ *or pediatrist in a department of hospital* $Hosp_2$.

Consider now the dynamic constraint $c_2 \equiv (RoleSet, 2)_a$. *Without hierarchies, the meaning of this constraint is that of preventing an individual to be active as a doctor in both the specified hospitals* $Hosp_1$ *and* $Hosp_2$. *In presence of the role-schema hierarchy pointed out in Figure 5, the meaning of this constraint becomes that of preventing an individual to be active as a doctor in hospital* $Hosp_1$ *or pediatrist in a department of hospital* $Hosp_1$ *and at the same time be active as doctor in hospital* $Hosp_2$ *or pediatrist in a department of hospital* $Hosp_2$. ◇

8. PROPERTIES OF CONSTRAINED GEO-RBAC

In the previous section, we have introduced several classes of GEO-RBAC constraints. The main problem in dealing with constraints is the complexity of their management. Even if the investigation of administrative operations against GEO-RBAC is outside the goals of this paper, we believe that a clear identification of the expressivity and the complexity of the proposed constraints is a very relevant issue in order to establish the usability of the proposed model. To this purpose, in the following we present several properties concerning GEO-RBAC and GEO-HRBAC constraints (hereinafter simply called GEO-RBAC constraints). Some of the presented properties extend already known results to the new classes of constraints. Others represent new results concerning GEO-RBAC specific characteristics. In both cases, since SI and DIA constraints correspond to traditional static and dynamic SoD constraints, the formalization and the results we present are still valid for RBAC constraints. The proofs of the presented results are reported in the Appendix.

8.1 Preliminaries

In order to present GEO-RBAC properties, we rely on the following definitions:

—Any assignment to sets, relations, and functions of Figure 1, except SUA, $SessionUsers$, $SessionRoles$, $EnabledSessionRoles$, is called *GEO-RBAC instance*.

—Given a GEO-RBAC instance $\mathcal{I}$, any assignment to relations SUA is called *static status* of $\mathcal{I}$.

—Given a GEO-RBAC instance $\mathcal{I}$, any assignment to functions SUA, $SessionUsers$, $SessionRoles$, $EnabledSessionRoles$ is called *dynamic status* of $\mathcal{I}$.

We notice that the concept of GEO-RBAC instance refers to an assignment to the basic sets of the GEO-RBAC model. The concept of static status refers to a particular assignment of roles to users, which therefore does not change at run-time; on the other hand, a dynamic status refers to the information concerning sessions, activations, and enabled roles, which may change due to different user behaviour at run time. We notice that static constraints will be checked against static status whereas dynamic constraints have to be checked against dynamic ones.

The problems we investigate in the following can now be informally stated as follows:

(1) *Satisfiability*: is it possible to check whether a set of constraints is satisfiable for a given GEO-RBAC instance?
(2) *Implications between constraint classes*: given a class of constraints, based on the properties of constraints in such class, what other properties can be derived for constraints belonging to other classes?
(3) *Propagation along hierarchies*: given a constraint over a certain role set (role schema set), what constraints for roles (role schemas) that are father, children of the given ones can be derived from it?
(4) *Evaluation*: what is the complexity of checking a given (static, dynamic) constraint against a given GEO-RBAC instance (static, dynamic) status?

For the sake of simplicity, in the following each role instance is denoted by ri_j and each role schema by rs_j.

8.2 Satisfiability

Given a set of (static, dynamic) constraints I for a given GEO-RBAC instance $\mathcal{I}$, such set is *satisfiable* in $\mathcal{I}$ if there exists at least one (static, dynamic) status for $\mathcal{I}$ for which the constraint evaluation returns true for each constraint. Thus, satisfiability is an important property for ensuring that each set of constraints is meaningful for the instance for which it has been defined. When a constraint is true over any status of a given GEO-RBAC instance, we say that it is *valid*. Thus, valid constraints do not need to be checked when the status changes. More formally, constraint satisfaction and validity can be defined as follows.

Definition 8.1 *Constraint Satisfiability and Validity*. Le c be a (static, dynamic) GEO-RBAC constraint for a GEO-RBAC instance $\mathcal{I}$. Let s be a (static, dynamic) status for $\mathcal{I}$. We say that c is true in s, denoted by $s \models c$ if the evaluation of c over s returns true We say that c is satisfiable in $\mathcal{I}$, denoted by $\mathcal{I} \models_s c$, if there exists at least one (static, dynamic) GEO-RBAC status s for $\mathcal{I}$ such that $s \models c$. We say that c is valid in $\mathcal{I}$, denoted by $\mathcal{I} \models c$, if for any (static, dynamic) GEO-RBAC status s for $\mathcal{I}$, $s \models c$ holds. A set of (static, dynamic) GEO-RBAC constraints $C = \{c_1, ..., c_n\}$ is satisfiable in $\mathcal{I}$ if there exists at least one GEO-RBAC (static, dynamic) status s for $\mathcal{I}$ such that $\mathcal{I} \models_s c_i, i = 1, ..., n$. C is valid in $\mathcal{I}$ if $\mathcal{I} \models c_i, i = 1, ..., n$. ◇

The first result we present points out a condition under which DIE constraints are valid. The proposed condition concerns the extents of the roles over which the constraint is defined. Indeed, it is easy to show that when the extents of those roles have an empty intersection, the constraint is obviously satisfied, since no position exists for which all the

roles can be enabled. We notice that, even if such roles cannot be concurrently enabled, and thus the constraint considered at enabling time is useless, this is not the case when it is considered at activation time. Indeed, if extents are disjoint, the roles can however be concurrently activated at different times and at different user positions.

THEOREM 8.2. *Let* $c = (RoleSet, n)_e \in DIE$ *be a constraint for a GEO-RBAC instance* $\mathcal{I}$. *If* $\bigcap_{ri_j \in RoleSet} ri_j.e = \emptyset$, *c is valid in* $\mathcal{I}$. ◇

A similar consideration holds for DSNSE and DSSE constraints, when the intersection of the union of the role extents, instances of the considered role schemas, is empty. Thus, the following result can be proved as a corollary of Theorem 8.2.

COROLLARY 8.3. *Let* $c_1 = (\{rs_1, ..., rs_m\}, n)_e \in DSNSE$ *and* $c_2 = (rs_1, rs_2, rel)_e \in DSSE$ *be two constraints for a GEO-RBAC instance* $\mathcal{I}$. *Let* $E_i = \bigcup_{ri_j \in Ext(rs_i)} ri_j.e$, $j = 1, ..., m$. *If* $\bigcap_{i=1,...,m} E_i = \emptyset$, c_1 *is valid in* $\mathcal{I}$. *If* $E_1 \bigcap E_2 = \emptyset$, c_2 *is valid in* $\mathcal{I}$. ◇

As we saw before, valid constraints are always true, thus there is no need to check them. Another particular case is represented by *unsatisfiable* constraints, i.e., constraints for which no status exists in which they are true. Such constraints represent badly defined requirements and should be removed from the constraint set. The unsatisfiability condition we present concerns GEO-RBAC constraints defined over hierarchically organized roles or role schemas. Indeed, it is easy to prove that, given a GEO-RBAC constraint (S, n), when at least n roles/role schemas in S have a descendant in common, since authorized users are inherited from descendant to ancestor roles, the constraint is unsatisfiable. The next result extends those presented in [Strembeck 2004] to the GEO-RBAC framework.

THEOREM 8.4. *Let* $c = (S, n)_f$, $f \in \{\perp, a, e\}$, *be a GEO-RBAC constraint for a GEO-RBAC instance* $\mathcal{I}$. *Let* $t = i$ *and* $T = R_I$ *if the constraint is instance-based and* $t = s$ *and* $T = R_S$ *if the constraint is schema-based. Let* $\preceq_t^*$ *denote the transitive closure of* $\preceq_t$. *If there exists* $s_1, ..., s_n \in S$ *and* $s \in T$ *such that* $s_i \preceq_t^* s$, $i = 1, ..., n$, *then c is unsatisfiable in* $\mathcal{I}$. ◇

Theorem 8.4 specifies conditions under which the corresponding constraint is never satisfied. Thus, it seems reasonable to assume that all constraints defined for a given GEO-RBAC instance do not satisfy such conditions. Under this assumption, we can prove that all remaining GEO-RBAC constraints are always satisfiable, as shown by the following theorem.

THEOREM 8.5. *A set of GEO-RBAC constraints for a GEO-RBAC instance* $\mathcal{I}$ *do not satisfying the condition pointed out in Theorem 8.4, are always satisfiable in* $\mathcal{I}$. ◇

8.3 Implications between constraint classes

The proposed classes of constraints are not independent. Indeed, it can be shown that when a certain constraint is true over a given GEO-RBAC status, other constraints, belonging to other classes, are also true. Similar properties hold for satisfiability and validity. Stated in another way, it is possible to show that some constraints *imply* other constraints. The concept of implication can be formalized as follows.

Definition 8.6 *Constraint implication*. Let C be a set of GEO-RBAC constraints for a GEO-RBAC instance $\mathcal{I}$. Let $c_1, c_2 \in C$. We say that c_1 implies c_2, denoted by $c_1 \Rightarrow c_2$, if for any GEO-RBAC status s for $\mathcal{I}$ such that $s \models c_1$, $s \models c_2$ holds. ◇

Implication is a very important property in order to avoid the specification of redundant constraint, i.e., constraints that are implied by other already existing constraints. Redundant constraints can be removed from the constraint set, thus reducing the time for constraint checking. In the following, for each aspect we have considered, i.e., granularity (schema/instance level), dimension (spatial/non-spatial), and verification time (static/dynamic at activation time/dynamic at enabling time), several interesting implications will be proved.

The results we are going to present rely on the concept of *corresponding* constraints. Informally, two constraints are corresponding if they define the same sets of mutually exclusive roles or role schemas, by changing either the granularity, dimension, or verification time.

Definition 8.7 *Corresponding constraints*. Let $c_1 = (S_1, n_1)_{f_1}$, $c_2 = (S_2, n_2)_{f_2}$, $c_3 = (RS_1, RS_2, rel)_{f_3}$ be GEO-RBAC constraints for a GEO-RBAC instance $\mathcal{I}$. c_1 and c_2 are corresponding if one of the following conditions hold:

—$S_1 = S_2$, and $n_1 = n_2$;

—c_1 is schema-based, c_2 is instance-based, $|S_1| \geq 2$, $n_1 = n_2$, and for all $rs_j \in S_1$ there exists exactly one role instance $ri_j \in S_2$ such that $ri_j \in Ext(rs_j)$;

—c_1 is schema-based, c_2 is instance-based, $S_1 = \{rs\}$, $n_1 = n_2$, $S_2 \subseteq Ext(rs)$.

c_1 and c_3 are corresponding if $S_1 = \{rs_1, rs_2\}$ and $n_1 = 2$. ◇

The relationships that can be proved can be summarized as follows (the proofs of such results trivially follows from the constraint definitions):

—Constraints defined at schema level implies a set of corresponding constraints at instance level, defined for each combination of instances of the role schemas it refers to; thus, the usage of constraints at schema level is a more convenient way for specifying constraints that hold for all role schema instances.

—Based on the fact that if n roles cannot be statically assigned to any user they cannot be simultaneously activated by any user, and therefore, they cannot be simultaneously enabled for any user, it is easy to show that each static constraint implies the corresponding dynamic constraint at activation time, which then implies the corresponding constraint at enabling time. This means that it is useless to specify corresponding constraints at different verification time. Indeed, at most one is sufficient to specify a given separation of duty constraint.

—Non-spatial schema constraints involving only two role schemas implies a set of corresponding spatial constraints, one for each topological relation. Thus, when a non-spatial constraint has already been specified, any spatial constraint corresponding to it is useless.

PROPOSITION 8.8 IMPLICATIONS BASED ON GRANULARITY. *Let c_1 be a schema-based non-spatial GEO-RBAC constraint. Let c_2 be an instance-based GEO-RBAC constraint corresponding to c_1. Then, $c_1 \Rightarrow c_2$.* ◇

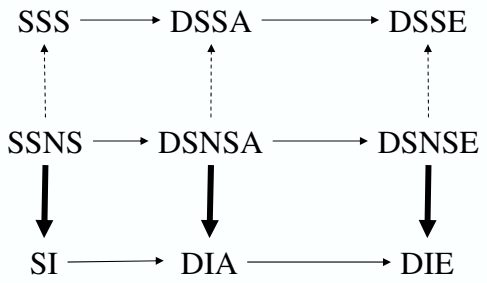

Fig. 7. Class implications for corresponding constraints

PROPOSITION 8.9 IMPLICATIONS BASED ON VERIFICATION TIME. *Let c_1 be a static GEO-RBAC constraint. Let c_2 be the dynamic constraint at activation time corresponding to c_1. Let c_3 be the dynamic constraint at enabling time corresponding to c_1. Then, $c_1 \Rightarrow c_2 \Rightarrow c_3$.* ◇

PROPOSITION 8.10 IMPLICATIONS BASED ON DIMENSION. *Let c_1 be a non-spatial GEO-RBAC constraint. Let c_2 be spatial GEO-RBAC constraint corresponding to c_1. Then, $c_1 \Rightarrow c_2$.* ◇

Implications pointed out by Propositions 8.9, 8.8, and 8.10 are graphically represented in Figure 7, where an arrow from a class of constraints to another means that each constraint in the first class implies at least one constraint of the other. Simple arrows represent implication based on verification time, dashed arrow represent implications based on dimension, and bold arrows represent implication based on granularity.

As a final consideration, we remark that, besides the implications pointed out above, several other implications exist concerning constraints belonging to the same class. In general, the definition of a correct and complete deduction system for GEO-RBAC constraints is outside the scope of the paper. We therefore leave its investigation to future work.

8.4 Propagation along hierarchies

In the following, similarly to what has been presented in [Kuhn 1997], we show that any GEO-RBAC constraint is propagated downwards the role or role schema hierarchy, namely it holds also for the descendant roles or role schemas. We first show that the property holds for instance-based constraints, then as a corollary, we prove that it also holds for schema-based constraints.

THEOREM 8.11. *Let $c = (\{ri_1, ..., ri_m\}, n)_f$, $f \in \{\perp, a, e\}$, be an instance-based constraint for a GEO-RBAC instance $\mathcal{I}$. Let $ri'_1, ..., ri'_m \in R_I$ such that $ri_j \preceq_i ri'_j$, $j = 1, ..., m$. Then: $(\{ri_1, ..., ri_m\}, n)_f \Rightarrow (\{ri'_1, ..., ri'_m\}, n)_f$.* ◇

COROLLARY 8.12. *Let $c_1 = (\{rs_1, ..., rs_m\}, n)_f$ and $c_2 = (rs_1, rs_2, rel)_f$, $f \in \{\perp, a, e\}$, be two schema-based constraint for a GEO-RBAC instance $\mathcal{I}$. Let $rs'_1, ..., rs'_m \in R_I$ such that $rs_j \preceq_s rs'_j$, $j = 1, ..., m$. Then:*

(1) $(\{rs_1, ..., rs_m\}, n)_f \Rightarrow (\{rs'_1, ..., rs'_m\}, n)_f$

(2) $(rs_1, rs_2, rel)_f \Rightarrow (rs'_1, rs'_2, rel)_f$. ◇

8.5 Evaluation

In order to be easily verifiable, GEO-RBAC constraints should be evaluated in PTIME. For non-spatial constraints, such result can be proved as follows:

—first, we show that non-spatial GEO-RBAC constraints can be represented in FO^k (first-order logic with at most k variables);

—since evaluation of FO^k is in PTIME, as shown in [Kolaitis and Vardi 1998], we can conclude that evaluation of non-spatial GEO-RBAC constraints is in PTIME.

For SI constraints, the result we are going to prove has already been proved in [Li et al. 2004]. Thus, next theorem extends the results in [Li et al. 2004] to cope with all classes of GEO-RBAC constraints. As a consequence, it holds for SI and DIA constraints, corresponding to traditional static and dynamic SoD constraints [Ferraiolo et al. 2001].

THEOREM 8.13. *Non-spatial GEO-RBAC constraints can be represented in FO^k.* ◇

From the fact that evaluation and satisfaction of FO^k are in PTIME [Kolaitis and Vardi 1998], the following result follows.

THEOREM 8.14. *Non-spatial GEO-RBAC constraints can be evaluated in PTIME.* ◇

For spatial constraints, complexity results depend on the complexity of computing spatial properties between role extents. Such complexity depends on: (i) the type of spatial relationships; (ii) the type of objects used to model role extents. Concerning the object type, as pointed out in Section 3, each type of geometry compliant the OGC recommendations can be used to model a role extent. Concerning spatial relationships, we consider here topological relationships listed in Section 3, since they are the most used in practice.

It has been shown that, when input spatial objects can be represented as semi-algebraic sets,[1] topological relationships can be computed in PTIME [Forlizzi et al. 2003]. Spatial objects with linear boundary represent an example of objects that can be represented as semi-algebraic sets. Under this assumption, spatial GEO-RBAC constraints are tractable.

THEOREM 8.15. *When role extents corresponds to semi-algebraic sets, spatial GEO-RBAC constraints can be evaluated in PTIME.* ◇

9. CONCLUSIONS

In this paper we have presented GEO-RBAC, an extension of the RBAC model dealing with spatial and location-based information. Unlike other proposals of spatially aware access control models, GEO-RBAC relies on the OGC spatial model [Open GIS Consortium 1999] to model (spatial) objects, user positions, and geographically bounded roles, making the approach quite standard and flexible. Another important characteristic of the model is the ability to deal with either real positions, obtained from a given mobile terminal or a cellular phone, and logical ones, possibly represented at different granularities. By introducing the concept of role schema, the type of role extents and logical positions can be customized, depending on the function the role represents. Moreover, besides the concept of active role, the concept of enabled role has also been introduced, in order to determine the roles that are enabled in sessions based on user positions. In the paper we have also

[1] A semi-algebraic set is a subset of R^2 representing the solution set of a boolean combination of a set of real algebraic equations and inequalities.

extended GEO-RBAC with hierarchies supporting the inheritance of permissions and user assignment as well as activations between roles. We have also introduced constraints for GEO-RBAC. They extend classical separation of duty constraints to deal with the specific characteristics of GEO-RBAC, i.e., different granularity (schema/instance level), dimension (spatial/non-spatial), and verification time (static/dynamic at activation time/dynamic at enabling time). Several properties concerning satisfiability, implications, and evaluation of the proposed classes of constraints have also be presented, providing useful guidelines for constraint definition. Some of the presented properties extend some already known results to the new classes of constraints we have introduced. Others represent new results concerning GEO-RBAC specific characteristics.

Our work leaves room for many research directions. A first direction concerns the definition of specific administrative operations to deal with GEO-RBAC instances and design methodologies. In this context, the definition of a correct and complete deduction system for GEO-RBAC constraints is an important issue, in order to detect redundant constraints, thus reducing the number of constraints to be checked. Suitable methodologies, possibly based on current software engineering methodologies, need to be devised supporting the specification and maintenance of a GEO-RBAC role basis. A second important direction concerns the application of our model to sensor-based applications and pervasive computing environments. Here the main issues are related to the development of scalable highly distributed architectures and to techniques assuring the authenticity of location information. An important question is also related to user transactions in such environments. As users move, some roles are disabled and others are enabled. Also users may cross boundaries of administration domains, that can be characterized by different GEO-RBAC hierarchies. However, mappings may exist among such hierarchies, and therefore a user can be automatically assigned in a new domain a new role based on the role that he/she had in the previous domain. The implications of such dynamics on the management of transactions need to be investigated. Also interoperation strategies supporting multi-domain GEO-RBAC need to be developed.

REFERENCES

ATLURI, V. AND MAZZOLENI, P. 2002. A uniform indexing scheme for geo-spatial data and authorizations. In *Proceedings of the 16th IFIP WG 11.3 Conference on Data and Application Security*. Kluwer Academic Publishers, Cambridge, UK, 207–218.

BELUSSI, A., BERTINO, E., CATANIA, B., DAMIANI, M., AND NUCITA, A. 2004. An authorization model for geographical maps. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*. ACM Press, Washington DC, USA, 82–91.

BERTINO, E., DAMIANI, M., AND MOMINI, D. 2004. An access control system for a web map management service. In *Proceedings of the 14th International Workshop on Research Issues on Data Engineering (RIDE'04)*. IEEE Computer Society, Boston, USA, 33–39.

BHATTI, R., GHAFOOR, A., BERTINO, E., AND JOSHI, J. 2005. X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control. *ACM Transactions on Information and System Security (TISSEC) 8,* 2 (May), 187–227.

CHANDRAN, S. AND JOSHI, J. 2005. LoT RBAC: A location and time-based RBAC model. In *Proceedings of the 6th International Conference on Web Information Systems Engineering (WISE'05)*. Springer-Verlag, New York, USA, 361–375.

CHUN, S. AND ATLURI, V. 2000. Protecting privacy from continuous high-resolution satellite surveillance. In *Proceedings of the 14th IFIP 11.3 Annual Working Conference on Database Security*. Schoorl, The Netherlands, 233–244.

CLEMENTINI, E., FELICE, P. D., AND VAN OOSTEROM, P. 1993. A small set of formal topological relationships

suitable for end-user interaction. In *Proceedings of the 3rd International Symposium on Advances in Spatial Databases*. Lecture Notes in Computer Science, vol. 692. Springer-Verlag, Singapore, 277–295.

COVINGTON, M., LONG, W., SRINIVASAN, S., DEV, A., AHAMAD, M., AND ABOWD, G. 2001. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM symposium on Access control models and technologies (SACMAT'01)*. ACM Press, Chantilly, Virginia, USA, 10–20.

COVINGTON, M., MOYER, M., AND AHAMAD, M. 2000. Generalized role-based access control for securing future applications. In *Proceedings of the 23rd National Information Systems Security Conference*.

FERRAIOLO, D., SANDHU, R., GAVRILA, S., KUHN, D., AND CHANDRAMOULI, R. 2001. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC) 4,* 3 (August), 224–274.

FORLIZZI, L., KUIJPERS, B., AND NARDELLI, E. 2003. Region-based querz languages for spatial databases in the topological data model. In *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD'03)*. Lecture Notes in Computer Science, vol. 2750. Springer-Verlag, Greece, 344–361.

HANSEN, F. AND OLESHCHUK, V. 2003a. Spatial role-based access control model for wireless networks. In *Proceedings of the 58th IEEE Vehicular Technology Conference (VTC'03)*. Vol. 3. IEEE Computer Society, Orlando, USA, 2093–2097.

HANSEN, F. AND OLESHCHUK, V. 2003b. SRBAC: a spatial role-based access control model for mobile systems. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NORDSEC'03)*. Gjvik, Norway, 129–141.

JOSHI, J., BERTINO, E., LATIF, U., AND GHAFOOR, A. 2005. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering 17,* 1 (January), 4–23.

KOLAITIS, P. AND VARDI, M. 1998. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS'98)*. ACM Press, Seattle, Washington, USA, 205–213.

KUHN, D. 1997. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In *Proceedings of the 2nd ACM workshop on Role-based access control (RBAC'97)*. ACM Press, Fairfax, Virginia, USA, 23–30.

LI, N., BIZRI, Z., AND TRIPUNITARA, M. 2004. On mutually-exclusive roles and separation of duty. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*. Washington DC, USA, 42–51.

MATHEUS, A. 2005. Declaration and enforcement of fine grained access restrictions for a service-based geospatial data infrastructure. In *Proceedings of the 10th ACM symposium on Access control models and technologies (SACMAT'05)*. ACM Press, Stockholm, Sweden,, 21–28.

OPEN GIS CONSORTIUM. 1999. Open GIS simple features specification for SQL. Revision 1.1.

OPEN GIS CONSORTIUM. 2001. The open GIS abstract specification. topic 1: Feature geometry (ISO 19107 spatial schema). Version 5.

OPEN GIS CONSORTIUM. 2003. Open GIS geography markup language (GML) implementation specification. Version 3.00.

PUREVJII, B., MAGASA, T. A., IMAI, S., AND KANAMORI, Y. 2004. An access control model for geographic data in an XML-based framework. In *Proceedings of the 2nd International Workshop on Security In Information Systems (WOSIS'04)*. INSTICC Press, Porto, Portugal, 251–260.

SANDHU, R., COYNE, E., FEINSTEIN, H., AND YOUMAN, C. 1996. Role-based access control models. *IEEE Computer 29,* 2, 38–47.

SANDHU, R., FERRAIOLO, D., AND KUHN, D. 2000. The NIST model for role-based access control: towards a unified standard. In *Proceedings of the 5th ACM workshop on Role-based access control (RBAC'00)*. ACM Press, Berlin, Germany, 47–63.

STREMBECK, M. 2004. Conflict checking of separation of duty constraints in RBAC - implementation experiences. In *Proceedings of the Conference on Software Engineering (SE'04)*. Innsbruck, Austria, 224–229.

Received Month Year; revised Month Year; accepted Month Year

A. PROOFS OF THE PRESENTED RESULTS

PROOF OF PROPOSITION 5.5.

(1) Property 1 trivially follows from Definition 5.4.
(2) Property 2: the condition $r_{i_2} \in EnabledSessionRole^+(s, rp)$ implies that $r_{i_2} \in SessionRoles^+(s)$. Thus, based on Definition 5.4, $r_{i_1} \in SessionRoles^+(s)$. Moreover, since $r_{i_1} \preceq_i r_{i_2}$, the location computed for role r_{i_2} is contained in the location of role r_{i_1} therefore $r_{i_1} \in EnabledSessionRole^+(s, rp)$.
(3) Property 3: since $MaxIRole \cong< MaxS, Bot_f >$ and $LocObj(Bot_f) = \perp$, it follows that rp is never contained in the empty extent. ◇

PROOF OF THEOREM 5.7.

(1) $CheckMRS(p, o)$ implies $CheckAll(p, o)$ simply follows from the definition of most specific role.
(2) To demonstrate $CheckAll(p, o)$ implies $CheckMRS(p, o)$, we show that whenever (p, o) is a permissions assigned to an enabled role, then a most specific role (*msr*) exists that has assigned (p, o) . Suppose a role r exists with $r \in EnabledSessionRoles^+(s, rp)$ such that $(p, o) \in I\_PrmsAssignment^*(r)$. We can now have two cases. In the first case, r does not have descendants which are enabled and thus, by definition, r is a most specific role. If the opposite case holds, we proceed as follows. We consider the set $dset$ of roles which are immediate descendants of r. By definition these roles inherit all the permissions of r and thus also (p, o). Hence we check whether any of the roles in $dset$ is a *mrs*. If a *mrs* is found we stop, otherwise we iterate the process and consider the immediate descendants of roles in $dset$. Note that this process terminates at most when one of the descendants is MaxIRole. In fact, based on Proposition 5.5, MaxIRole is never enabled and thus the descendant of r which immediately precedes MaxIRole is a mrs. Thus a *mrs* always exists, that is what we wanted to demonstrate. ◇

PROOF OF THEOREM 8.2. The proof follows from the fact that, based on the hypothesis, a given real position may belong to at most one role extent (since they are disjoint and since, based on Definition 4.1, logical positions for a given role schema are always contained inside role extents). Thus, for all positions $pos \in RPOS$ and for all sessions $t \in SES$, $EnabledSessionRoles^+(t, pos) \subseteq \{ri_j\}$ for some $ri_j \in RoleSet$. Thus, the cardinality of $EnabledSessionRoles^+(t, pos)$ is always lower than n (since $n \geq 2$). We also notice that, under the hypothesis, $RoleSet$ cannot contain two roles ri_1 and ri_2 such that $ri_1 \preceq_i ri_2$. Indeed, in this case, according to Definition 5.1, $ri_2.e \subseteq ri_1.e$, thus the hypothesis cannot be satisfied. ◇

PROOF OF THEOREM 8.4. We present the proof for instance-based constraints ($SI \cup DIA \cup DIE$). The proof for schema-based constraints is almost similar. For instance-based constraints, s_i is a role, therefore in the following it is denoted by ri_j. If there exists $ri \in R_I$ such that $ri_j \preceq_i^* ri$, $j = 1, ..., n$, from Propositions 5.3 and 5.5, and Definition 5.4, the following facts hold:

(1) $I\_AuthorizedUsers(ri) \subseteq I\_AuthorizedUsers(ri_j)$, $j = 1, ..., n$
(2) for all $t \in SES$, if $ri \in SessionRoles^+(t)$, then $ri_j \in SessionRoles^+(t)$, $j = 1, ..., n$

(3) for all $t \in SES$ and $rp \in RPOS$, if $ri \in EnabledSessionRoles^{+}(t, rp)$, then $ri_j \in EnabledSessionRoles^{+}(t, rp)$, $j = 1, ..., n$.

From the previous facts it follows that:

(1) *SI constraints*: in Definition 7.1, $h = \{ri_1, ..., ri_n\}$ makes the formula of static constraints false, thus the constraint is unsatisfiable.

(2) *DIA constraints*: in Definition 7.4, consider a session t of a user u such that $u \in I\_AuthorizedUsers(ri)$. In this case, there exists a status in which $SessionRoles^{+}(t) = \{ri_1, ..., ri_n\}$. By setting $h = SessionRoles^{+}(t)$, the constraint is violated.

(3) *DIE constraints*: in Definition 7.4, consider a session t of a user u such that $u \in I\_AuthorizedUsers(ri)$. In this case, there exists a status in which $SessionRoles^{+}(t) = \{ri_1, ..., ri_n\}$. By setting $h = SessionRoles(t)$ and, by choosing $pos \in ri_e$, the constraint is violated. ◇

PROOF SKETCH OF THEOREM 8.5. From the constraint definition, it follows that when each user is assigned to at most one role instance all the constraints are satisfied. Since all constraints are defined over roles, or role schema, that do not have a common descendant, such assignment always exists and this proves the theorem. ◇

PROOF OF THEOREM 8.11. The proof is presented for SI constraints. The proof for DIA and DIE constraints is almost similar to the presented one. For the sake of simplicity, in the following, we assume that $m = 2$. Consider a static status s such that $s \models c$. Now, assume, per absurdo, that in s there exists a user u such that $u \in (SR\_AssignedUsers(ri'_1) \cap SR\_AssignedUsers(ri'_2))$. Since $ri_1 \preceq_i ri'_1$ and $ri_2 \preceq_i ri'_2$ then $u \in SR\_AssignedUsers(ri_1) \cap SR\_AssignedUsers(ri_2))$. Thus, $s \not\models c$, which leads to a contradiction. ◇

PROOF OF COROLLARY 8.12. The proof is presented for Property (1) and SSNS constraints; the proof for Property (2) and all the other constraint classes can be shown in a similar way. For the sake of simplicity, in the following, we assume that $m = 2$. Consider a static status s such that $s \models c$. Now, suppose that $s \not\models (\{rs'_1, rs'_2\}, n)_f$. This means that there exist $ri'_j \in Ext(rs'_j)$, $j = 1, 2$ and there exists a user u such that $u \in (SR\_AssignedUsers(ri'_1) \cap SR\_AssignedUsers(ri'_2))$. But since $rs_1 \preceq_i rs'_1$ and $rs_2 \preceq_i rs'_2$, this means that there exist $ri_j \in Ext(rs_j)$, $j = 1, 2$ such that $u \in SR\_AssignedUsers(ri_1) \cap SR\_AssignedUsers(ri_2))$. Thus, $s \not\models c$, which leads to a contradiction. ◇

PROOF SKETCH OF THEOREM 8.13. In the following, we present the proof for SSNS, and DIE constraints. All the other proofs can be obtained using a similar approach. In both cases, we assume that no hierarchies are available. Then, we discuss how such proof can be extended when hierarchies are available.

(1) *SSNS constraints*. Let $SSNS(n) \equiv (SchemaSet, n)_{\perp} \in SSNS$, with $SchemaSet = \{rs_1, ..., rs_m\}$. We consider the inclusion of a role schema rs in $SchemaSet$ as a predicate denoted by $SchemaSet(rs)$, the inclusion of a user u in U as a predicate denoted by $U(u)$, the fact that a role r is an instance of a role schema rs as a predicate denoted by $Ext(rs, r)$, and function $SR\_AssignedUsers$ as a predicate $SR\_AssignedUsers(u, r)$ which is true when $u \in SR\_AssignedUsers(r)$. In order to present the representation of $SSNS(n)$ in FO^k we consider two cases, depending on the value of n.

If $n \geq 2$, $SSNS(n)$ can be represented in FO^k as the following formula, denoted by $SSNS_2(n)$:

$$\forall rs_{i_1}, ..., rs_{i_n}[\bigwedge_{h,k=1,...,n;h\neq k} rs_{i_h} \neq rs_{i_k} \wedge SchemaSet(rsi_{i_1}) \wedge ... \wedge SchemaSet(rsi_{i_n}) \rightarrow$$

$$\neg\exists ri_{i_1}, ..., ri_{i_n}, u[Ext(rs_{i_1}, ri_{i_1}) \wedge ... \wedge Ext(rs_{i_n}, ri_{i_n}) \wedge$$

$$U(u) \wedge SR\_AssignedUsers(u, ri_{i_1}) \wedge ... \wedge SR\_AssignedUsers(u, ri_{i_n})]]$$

If $n = 1$, and denoting by rs the role schema in $SchemaSet$, we get the following formula, denoted by $SSNS_1(n)$:

$$\exists rs[SchemaSet(rs) \wedge \neg\exists ri_{i_1}, ..., ri_{i_n}, u[\bigwedge_{h,k=1,...,n;h\neq k} ri_{i_h} \neq ri_{i_k} \wedge Ext(rs, ri_{i_1}) \wedge ... \wedge$$

$$Ext(rs, ri_{i_n}) \wedge U(u) \wedge SR\_AssignedUsers(u, ri_{i_1}) \wedge ... \wedge SR\_AssignedUsers(u, ri_{i_n})]]$$

Thus, a generic non-spatial constraint at schema level $(SchemaSet, n)_{\perp}$ can be represented by a formula $SSNS(n)$ as follows:

$$[\exists rs_1, rs_2[SchemaSet(rs_1) \wedge SchemaSet(rs_2) \wedge rs_1 \neq rs_2 \wedge SSNS(n)]] \vee$$

$$[\nexists rs_1, rs_2[SchemaSet(rs_1) \wedge SchemaSet(rs_2) \wedge rs_1 \neq rs_2 \wedge SSNS(n)]]$$

The number of variables is $2n + 1$ in $SSNS_2(n)$ and $n + 2$ in $SSNS_1(n)$. In the first case, n is bounded by the cardinality of $SchemaSet$. In the second case, n is bound by the cardinality of $Roles$. Thus, $SSNS(n) \in FO^{max(2|SchemaSet|+3, |Roles|+4}$.
In presence of hierarchies, function $SR\_AssignedUsers$ (and therefore predicate $SR\_AssignedUsers$) has just to be replaced by function $I\_AuthorizedUsers$ (and therefore by a predicate $I\_AuthorizedUsers$). The same results hold.

(2) *DIE constraints*. Let $DIE(n) \equiv (RoleSet, n)_e \in DIE$, with $RoleSet = \{ri_1, ..., ri_m\}$. We model the inclusion of a role r in $Roleset$ as a predicate denoted by $Roleset(r)$, the inclusion of a sessione t in SES as a predicate denoted by $SES(t)$, the inclusion of a real position pos in $RPOS$ as a predicate denoted by $RPOS(pos)$, and function $EnabledSessionRoles$ as a predicate $ESR(t, pos, r)$ which is true when $r \in EnabledSessionRoles(t, pos)$. $DIE(n)$ can be represented in FO as follows:

$$\forall s \in SES\, \forall ri_{i_1}, ..., ri_{i_n}[\bigwedge_{h,k=1,...,n;h\neq k} ri_{i_h} \neq ri_{i_k} \wedge RoleSet(ri_{i_1}) \wedge ... \wedge RoleSet(ri_{i_n}) \rightarrow$$

$$\neg\exists pos[RPOS(pos) \wedge ESR(t, pos, ri_{i_1}) \wedge ... \wedge ESR(t, pos, ri_{i_n})]]$$

In the previous first-order formula, the number of variables is $n + 2$, and n is bounded by the cardinality of $RoleSet$. Thus, $DIE(n) \in FO^{|RoleSet|+2}$.
In presence of hierarchies, function $EnabledSessionRoles$ (and therefore predicate ESR) has just to be replaced by function $EnabledSessionRoles^+$ (and therefore by a new predicate ESR^+). The same results hold. ◇