Organization based access control

Anas Abou El Kalam[§] Rania El Baida[‡] Philippe Balbiani[‡*]
Salem Benferhat[♭] Frédéric Cuppens[‡*] Yves Deswarte[§]
Alexandre Miège[¶] Claire Saurel[¶] Gilles Trouessin[†]

Cril[†] Ernst & Young Audit[‡] Irit[§] Laas[¶] Onera[‖]

Abstract

None of the classical access control models such as DAC, MAC, RBAC, TBAC or TMAC is fully satisfactory to model security policies that are not restricted to static permissions but also include contextual rules related to permissions, prohibitions, obligations and recommendations. This is typically the case of security policies that apply to the health care domain. In this paper, we suggest a new model that provides solutions to specify such contextual security policies. This model, called Organization based access control, is presented using a formal language based on first-order logic.

1 Introduction

Several access control models have been proposed: DAC[1], MAC[2, 3], RBAC[4, 5, 6], TBAC[7] or TMAC[8]. None of these models is fully satisfactory, seeing that they are difficult to apply to organizations that use security policies including:

1. Rules that specify contextual permissions or prohibitions. There are many examples of rules that apply only to specific circumstances. For instance, in the health care domain, physicians have special permissions in specific contexts, such as the context of urgency (see section 5).

2. Rules that specify obligations or recommendations. Classical access control models are generally restricted to permissions. Some of them include prohibitions. More recently, including obligations in the security policy specification was also suggested[9, 10]. As far as we know, recommendations have never been considered previously.

3. Rules that are specific to the organization. In particular, the organization may be structured into several sub-organizations having their own security policy. The model should provide means to specify the different security policies within a unique framework.

This paper suggests a model that attempts to solve these problems. The concept of organization is central in this model. The specification of the security policy is completely parameterized by the organization so that it is possible to handle simultaneously several security policies associated with different organizations. The model is not restricted to permissions, but also includes the possibility to specify prohibitions, obligations and recommendations.

The remainder of this paper is organized as follows. Sections 2 and 3 present basic models for discretionary access control and role based access control. Section 4 presents our new model, the "Organization based access control" (ORBAC). Section 5 defines a language based on first-order logic that will be used to specify an ORBAC security policy. Section 6 develops an example of security policy in this language. Section 7 shows how to specify various constraints. Finally section 8 concludes the paper. This work is supported by the RNRT project MP6.

*Corresponding author: cuppens@irit.fr.

[†]Cril-CNRS, Centre de recherche en informatique de Lens, Université d'Artois, Rue Jean Souvraz, SP18, 62307 Lens Cedex.

[‡]Ernst & Young Audit, 1 place Alphonse Jourdain, 31000 Toulouse.

[§]Irit-CNRS, Institut de recherche en Informatique de Toulouse, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex 4.

[¶]Laas-CNRS, Laboratoire d'analyse et d'architecture des systèmes, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4.

[‖]Office national d'études et de recherches aérospatiales, Centre de Toulouse, BP 4025, 31055 Toulouse Cedex 4.

2 Discretionary access control

Harrisson, Ruzzo and Ullman[11] define a security policy model, the HRU model, that applies to subjects, objects and actions. Intuitively, a subject is an active entity. In the context of an information system, it includes the users of this system and, generally, the processes that run on behalf of these users. The set of objects contains the set of all active entities. It also includes non-active entities. For instance, in an information system, files and directories are objects. Actions enable the subject to have direct access to objects. They generally correspond to elementary actions such as reading a file or writing a file. In this model, the security policy is limited to the specification of permissions, i.e. relationships between subjects, objects and actions. These relationships are generally represented by a matrix A of permissions. If s is a subject and o is an object then $A(s, o)$ represents the set of actions α that subject s is permitted to execute on object o. In the HRU model, it is necessary to enumerate, through a matrix of permissions, all the triples $\langle s, o, \alpha \rangle$ that correspond to a permission granted by the security policy. When new subjects, new objects or new actions are introduced in the system, it is necessary to update the security policy in order to record the permissions granted to these new entities. As a result, the updating of security policies specified according to this model become quite complicated. Another limit of the HRU model is that it only enables the administrator to specify permissions. Neither prohibitions, obligations nor recommendations are included.

3 Role based access control

In the Role based access control model, the RBAC model, the security policy does not directly grants permissions to users but to roles[5]. A given user will obtain permissions by playing roles, in which case this user will inherit all the permissions associated with these roles. It is possible to refine this model by including the concepts of session and role hierarchy. Within a session, a user is not obliged to activate all his/her roles but only the subset of his/her roles that are necessary to perform a given task. The role hierarchy is useful because permissions are inherited through this hierarchy. This simplifies the security policy specification. In the complete model, it is also possible to specify constraints. For instance, we may specify that there is no user that can play both roles anaesthetist and surgeon.

The RBAC model has the following drawbacks. First, the concept of permission is primitive. When specifying the security policy of a given application, the RBAC model must be refined to make explicit the structure of permissions. It is argued that this is because this structure might depend on the application. We consider that it would be better to include a generic structure of permissions in the model. In our model, permissions as well as prohibitions, obligations and recommendations are represented by relationships. Second, the concept of role hierarchy is not free of ambiguity. In particular, it is generally incorrect to consider that it corresponds to an organizational hierarchy. For instance, the director of an hospital is the head of the physicians of this hospital. However, director is an administrative position and it is not required to be a physician to hold this position. Therefore, it would be incorrect to consider that the role director inherits all the permissions of the role physician. This has been acknowledged by other authors, including the possible need to maintain different hierarchies. Third, the distinction between the concepts of role and group is not clear. Group is a concept that was used before the definition of the RBAC model and there were many discussions about the difference between access control models based on user groups and RBAC. The ORBAC model suggested in section 4 attempts to clarify this point.

In other respect, it is not possible, in the RBAC model, to specify a permission that depends on a given context. More precisely, if a given permission is granted to a given role, then all users that play this role will inherit the given permission. Therefore, it is not possible to specify that a physician is permitted to have a direct access to the patient records, unless he/she is one of the physician's patient[12, 13]. Moreover, as mentioned in the previous section, another limit of the RBAC model is that it only enables the administrator to specify permissions. Finally, additional limits appear when the RBAC model is used to specify the security policy of a system that includes several organizations.

4 Organization based access control

To overcome their limitations, several authors have recently proposed improved versions of these access control models. Some models include temporal constraints and support the periodic activation of roles[14, 15]. Other models considers that each organization has to define its own internal security policy while

respecting the constraints imposed by the global security policy[16]. And finally there are models based on the notion of coalitions, i.e., sets of organizations that collaborate together to fulfil their missions [**?**]. In this paper, we try to overcome the limitations of these access control models by considering the concept of organization together with the concept of context. In this section, we present our ORBAC model using a diagrammatic language based on the entity-relationship model. A presentation of our ORBAC model using a formal language based on first-order logic will be proposed in section5. In accordance with the entity-relationship model, the entities and the relationships of our ORBAC model may be associated with attributes. Since these attributes are generally specific to a particular application, we do not include them in the diagrammatic presentation.

4.1 Organizations

The most important entity in our model is the entity *Organization*. In the health care domain, we can take the following organizations: "Languedoc private clinic", the "casualty department of Purpan hospital", the "intensive care unit of Rangueil hospital", etc. Roughly speaking, an organization can be seen as an organized group of active entities, i.e. subjects, playing some role or other. Notice that a group of subjects does not necessarily correspond to an organization. More precisely, the fact that each subject plays a role in the organization corresponds to some agreement between the subjects to form an organization.

4.2 Subjects and roles

The entity *Subject* is used differently from one security model to another. In our ORBAC model, a subject will be either an active entity, i.e. a user, or an organization. Examples of subjects therefore include users such as "John", "Mary", "Peter", etc, or organizations such as the "accounts department of Languedoc private clinic", the "casualty department of Purpan hospital", the "intensive care unit of Rangueil hospital", etc. In our model, the entity *Role* is used to structure the link between subjects and organizations. In the health care domain, the roles "cardiologist", "nurse", "physician", etc, will be played by users whereas the roles "casualty department", "rescue team", "intensive care unit", etc, will be played by organizations. Seeing that subjects play roles in organizations, we need a relationship that joins up these entities together: the relationship *Employ* (see figure 1). If org is an organization, s a subject and r a role, then $Employ(org, s, r)$ means that org employs subject s in role r. Unlike the TMAC model or the

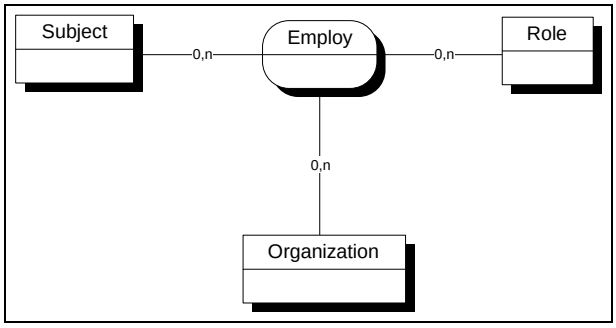

Figure 1: The *Employ* relationship

RBAC model which consider binary relations between organizations and subjects or between subjects and roles, notice that our model consider a ternary relation between organizations, subjects and roles. Again, let us remark that subjects might be users as well as organizations. To illustrate the truth of this, one has only to mention the examples:

- $Employ(Purpan, John, cardiologist)$: "the Purpan hospital employs John as a cardiologist" and
- $Employ(Rangueil, ICU31, intensive\_care\_unit)$: "the Rangueil hospital employs ICU31 as an intensive care unit".

4.3 Objects and views

In our model, the entity *Object* will mainly cover inactive entities such as data files, emails, printed forms, etc. In the health care domain, we will also have to consider objects like administrative records, medical records and surgical records of patients. By means of the entity *Role*, we will be able to structure the subjects and to update easily security policies when new subjects are added to the system. Seeing that we will also have to structure the objects and to add new objects to the system, we believe that a similar entity regarding objects is needed: the entity $View$. Roughly speaking, as in relational databases, a view corresponds to a set of objects that satisfy a common property. For instance, in a file management system, the view "administrative record" corresponds to the administrative records of patients whereas the view "medical record" corresponds to the medical records

of patients. Seeing that views characterize the ways objects are used in organizations, we need a relationship that links together these entities: the relationship Use (see figure 2). If org is an organization, o is an object and v is a view, then $Use(org, o, v)$ means that org uses object o in view v. We wish to focus the attention

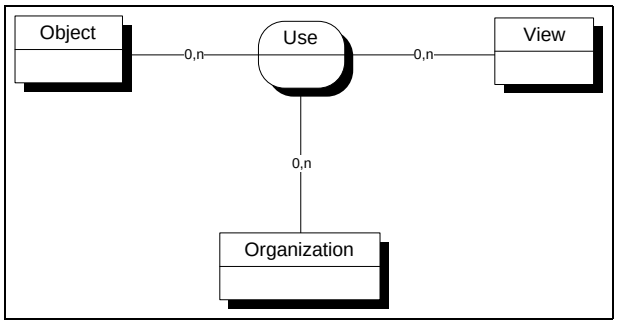

Figure 2: The Use relationship

on the fact that our model considers a ternary relation between organizations, objects and views. Our aim is to make ourselves able to characterize organizations that give different definitions to the same view. Take the case of the view "medical record" defined in Purpan hospital as a set of Word documents and defined in Rangueil hospital as a set of Latex documents:

- $Use(Purpan, F31.doc, medical\_record)$: "the Purpan hospital uses F31.doc as a medical record" and
- $Use(Rangueil, F32.tex, medical\_record)$: "the Rangueil hospital uses F32.tex as a medical record".

4.4 Actions and activities

Security policies specify the authorized accesses to inactive entities by active entities and regulate the actions carried out in the system. In our model, the entity *Action* will mainly contain computer actions such as "read", "write", "send", etc. Following the line of reasoning suggested in sections 4.2 and 4.3 where subjects and objects were abstracted by means of roles and views, a new entity will also be used to abstract actions: the entity *Activity*. Seeing that roles associate subjects that fulfil the same functions and views correspond to sets of objects that satisfy a common property, activities will join actions that partake of the same principles. In our model, activities like "reading", "writing", "consulting", etc, will be of the utmost interest. Since different organizations may decide that one and the same action comes under distinct activities, the relationship $Consider$ (see figure 3) will be used to join up the entities *Organization*, *Action* and *Activity*. More precisely, if org is an organization, α is an action and a is an activity, then $Consider(org, \alpha, a)$ means that org considers that action α falls within the activity a. Let

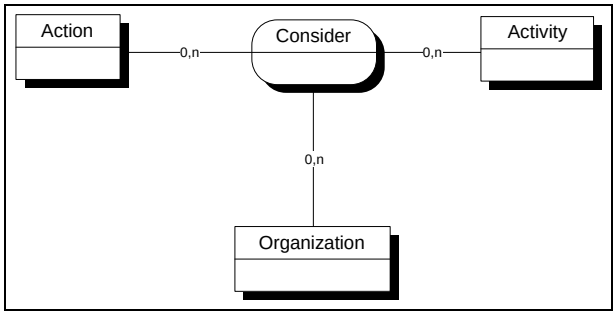

Figure 3: The $Consider$ relationship

us remark again that $Consider$ is a ternary relation between organizations, actions and activities. What we have in mind is to be able to characterize organizations that differently structure the same activities. We should consider, for instance, that activity "consulting" corresponds, in Purpan hospital, to an action "read" that can be ran on data files whereas it corresponds, in Rangueil hospital, to action "select" that can be performed on relational databases:

- $Consider(Purpan, read, consulting)$: "the Purpan hospital considers read as a consulting" and
- $Consider(Rangueil, select, consulting)$: "the Rangueil hospital considers select as a consulting".

4.5 Security policy (first definition)

Using the entities and the relationships introduced in the previous sections, we are now in a position to define security policies applying to such or such organization. A security policy specifies the authorized accesses of a system through a set of permissions, prohibitions, obligations and recommendations. In the following discussion, we consider only the concept of permission, given that similar arguments can be developed regarding the concepts of prohibition, obligation and recommendation. What we have in mind is to extend our model with a relationship *Permission* for the purpose of being

able to join together organizations, roles, views and activities. More precisely, if org is an organization, r is a role, v is a view and a is an activity, then $Permission(org, r, v, a)$ means that organization org grants role r permission to perform activity a on view v. For example, take the case of Purpan hospital granting role "medical secretary" permission to perform activity "creation" on view "administrative record", a security requirement expressed by the following fact $Permission(Purpan, medical\_secretary, administrative\_record, creation)$. However, the security requirement expressed by $Permission(Purpan, physician, medical\_record, consulting)$ says that Purpan Hospital grants role "physician" permission to perform activity "consulting" on view "medical record". It is highly likely that this security requirement is not exactly what Purpan hospital wants to specify: in normal circumstances, a physician is permitted to consult the contents of only those patient medical records for which he is the attending physician. The truth of the matter is that the ORBAC model described above simply cannot cope with such security requirement: given an organization, users inherit permissions from the roles they play in that organization. To solve this problem, we propose to extend our model with a new entity.

4.6 Contexts

Roughly speaking, contexts will be used to specify the concrete circumstances where organizations grant roles permissions to perform activities on views. In the health care domain, our new entity $Context$ will cover circumstances such as "urgency", "industrial medicine", "attending physician", etc. Every context can be seen as a ternary relation between subjects, objects and actions defined within such or such some organization. Therefore, entities $Organization$, $Subject$, $Object$, $Action$ and $Context$ are linked together by the relationship $Define$ (see figure 4): if org is an organization, s is a subject, o is an object, α is an action and c a context, then $Define(org, s, o, \alpha, c)$ means that within organization org, context c is true between subject s, object o and action α. The conditions required for a given context to be linked, within a given organization, to subjects, objects and actions will be formally specified by logical rules. This issue will be addressed in section 5. In the meantime, let us consider the following facts: $Define(Purpan, John, F31.doc, read, urgency)$ and $Define(Rangueil, Mary, F32.tex, read, attending\_physician)$. If the former fact is true then there is no need for John to be the attending physician of the patient concerned by medical record F31.doc: in circumstances such as "urgency", physicians have a direct access to medical records. If the latter fact is true then Mary must be the attending physician of the patient concerned by medical record F32.tex: in normal contexts such as "attending physician", physicians are permitted to consult the contents of only those patient medical records for which they are the attending physician.

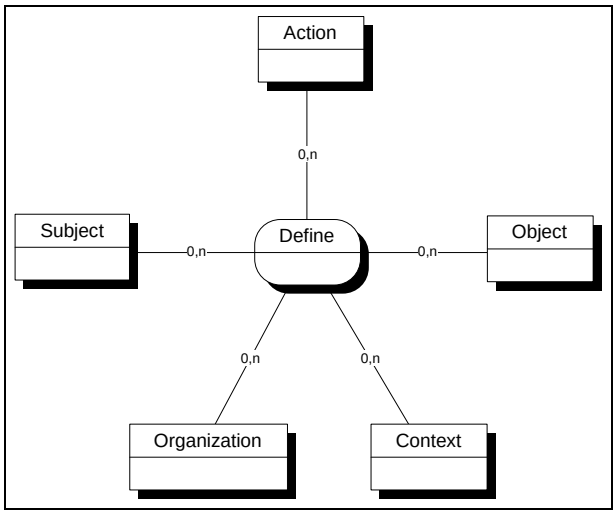

Figure 4: The $Define$ relationship

4.7 Security policy (final definition)

In view of our new entity $Context$, we can now give the final definition of security policies applying to such or such organization. The relationship $Permission$ now corresponds to a relation between organizations, roles, views, activities and contexts. The relationships $Prohibition$, $Obligation$ and $Recommendation$ are defined similarly (see figure 5). If org is an organization, r is a role, v is a view, a is an activity and c a context then $Permission(org, r, v, a, c)$ means that organization org grants role r permission to perform activity a on view v within context c. For instance, the security policy of Purpan hospital may include the facts $Permission(Purpan, physician, medical\_record, consulting, urgency)$ "Purpan hospital grants physicians permission to consult medical records within the context urgency" and $Permission(Purpan, physician, medical\_record, consulting, attending\_physician)$ "Purpan hospital grants physicians permission to consult medical records within the context attending physician".

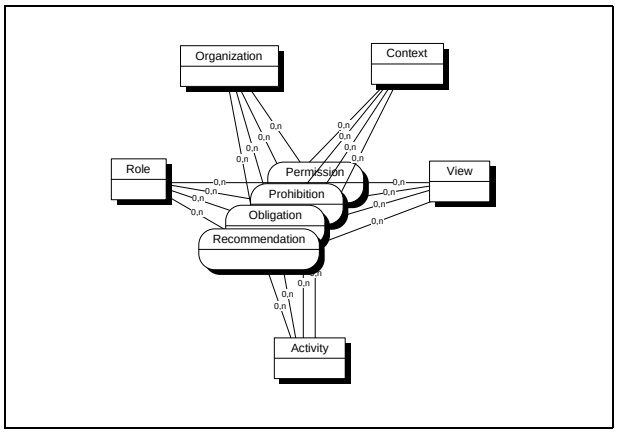

Figure 5: The *Permission*, *Prohibition*, *Obligation* and *Recommendation* relationships

4.8 Concrete authorization

The relationship *Permission* enables a given organization to specify permissions granted in a given context. Such permissions correspond to a ternary relation between roles, views and activities. However, down-to-earth access control must provide a framework for describing the concrete actions that may be performed by subjects on objects. For the purpose of modelling concrete permission, we introduce in our model the relationship $Is\_permitted$ as a relationship between subjects, objects and actions, i.e. if s is a subject, o is an object and α is an action then $Is\_permitted(s, o, \alpha)$ means that subject s is permitted to perform action α on object o. Relationships such as $Is\_prohibited$, $Is\_obliged$ and $Is\_recommended$ could be defined in the same way. Our relationship $Is\_permitted$ is similar to the notion of permission suggested in the model HRU. There is, however, a difference of major importance. In the original model HRU, each authorized triple $\langle s, o, \alpha \rangle$ must be explicitly stated, although more expressive discretionary access control models where permissions are logically derived have been proposed. In our model, triples that are instances of the relationship $Is\_permitted$ are logically derived from permissions granted to roles, views and activities by the relationship *Permission*. This is modelled by a general rule that will be presented in section 5. Explicit instances of the relationship $Is\_permitted$ may be viewed as exceptions to the general security policy specified by the relationship *Permission*.

Figure 6 resumes our security model. It contains 8 entities (*Organization*, *Subject*, *Role*, *Object*, $View$, *Action*, *Activity* and *Context*) and 12 relationships (*Employ*, Use, *Consider*, *Permission*, *Prohibition*, *Obligation*, *Recommendation*, $Is\_permitted$, $Is\_prohibited$, $Is\_obliged$, $Is\_recommended$ and $Define$).

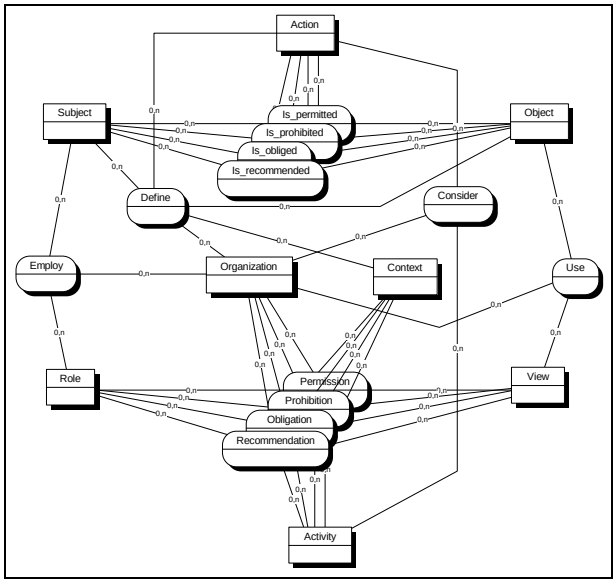

Figure 6: The ORBAC model

5 Language and axiomatical presentation

The purpose of this section is to present a logical framework that provides a means of representation and reasoning about permissions, prohibitions, obligations and recommandations in a given universe of entities. What we have in mind is to associate a first-order language $\mathcal{L}$ to the entity-relationship diagram described above. Our first-order language will provide a syntax capable of expressing precise statements about the relationships holding between entities. Each expression of $\mathcal{L}$ will contain symbols from a particular limited vocabulary separated into four groups: constant symbols, individual variables, relation symbols and function symbols.

The constant symbols will correspond to the instances of the entities of our diagram. As a result, there will be as many types θ of constant symbols as there are entities in our diagram, i.e. 8: *Organization*, *Subject*, *Object*, *Action*, *Role*, $View$, *Activity* and *Context*. For instance, we will have: constant symbols of type

Organization like *Purpan*, *Rangueil*, $ICU31$, etc, constant symbols of type *Subject* like *Jean*, *Mary*, $ICU31$, etc, constant symbols of type *Object* like $F31.doc$, $F32.doc$, $F33.tex$, etc, constant symbols of type *Action* like *read*, *write*, *consult*, etc, constant symbols of type *Role* like *physician*, *nurse*, *intensive_care_unit*, etc, constant symbols of type *View* like *administrative_record*, *medical_record*, *surgical_record*, etc, constant symbols of type *Activity* like *reading*, *writing*, *consulting*, etc. Constant symbols will be denoted by lower case Latin letters like a, b, c, etc.

Similarly, there will be individual variables for each type θ. They will be denoted by lower case Latin letters like x, y, z, etc. For all types θ, constant symbols of type θ and individual variables of type θ will also be called θ-terms.

As for the relation symbols of $\mathcal{L}$, denoted by capital Latin letters P, Q, R, etc, they will correspond to the 12 relationships of our diagram. Each relation symbol P of $\mathcal{L}$ is assumed to be a typed relation. More precisely: *Employ* is a relation symbol of type $(Organization, Subject, Role)$, Use is a relation symbol of type $(Organization, Object, View)$, *Consider* is a relation symbol of type $(Organization, Action, Activity)$, *Permission*, *Prohibition*, *Obligation* and *Recommendation* are relation symbols of type $(Organization, Role, View, Activity, Context)$, $Is\_Permitted$, $Is\_Prohibited$, $Is\_Obliged$ and $Is\_Recommanded$ are relation symbols of type $(Subject, Object, Action)$ and $Define$ is a relation symbol of type $(Organization, Subject, Object, Action, Context)$.

Using terms and relations, we can introduce the atomic formulas of $\mathcal{L}$ as follows: if t_1 is a θ_1-term, ..., t_n is a θ_n-term and P is a relation symbol of type $(\theta_1, \ldots, \theta_n)$ then $P(t_1, \ldots, t_n)$ is an atomic formula. Examples of atomic formulas are $Employ(Purpan, John, physician)$ and $Permission(Rangueil, medical\_secretary, administrative\_record, creation, normal)$.

At this point, our language is not expressive enough to capture the possibility to compare entities. In many applications, we are given a universe of entities and we wish to derive information about some of their properties. From a formal point of view, function symbols will be of use to us for describing or defining these entities. Function symbols will be denoted by lower case Latin letters like f, g, h, etc. Each function symbol f corresponds to some attribute, hence it is a typed function and it has an associated range V_f. The type and the range of a function symbol depend on the nature of the attribute it corresponds to. If a function symbol corresponds to the attribute *Name*, then it is of type *Subject* and its range is a set of names. Similarly, a function symbol corresponding to the attribute *Age* will be of type *Subject* and its range will be the set of all positive integers. Take another example: if a function symbol must correspond to the attribute *Blood_group*, then it is of type *Subject* and its range is $\{A, AB, B, O\}$. Since there might exist nameless subjects or subjects the blood group of which is not known, then the function symbols of $\mathcal{L}$ will only represent partial mappings between entities and associated ranges. In many real situations, we are not able to assign a single value of an attribute to an entity. To deal with such situations on a conceptual level, we will use unary function symbols the associated ranges of which are power sets. To illustrate the truth of this, one has only to mention the attribute *Attending_physician*: the associated function symbol is of type *Subject* and the associated range is a set of finite sets of names.

In order to draw conclusions from the partial information represented by function symbols, we need to introduce concrete binary relations, denoted by σ, τ, μ, etc, between domains. The type of a concrete binary relation is the pair consisting of the domains it corresponds to. Equality is probably the simplest concrete binary relation we will have to consider. We should consider the following examples:

- If t and u are terms of type *Subject*, then $Attending\_physician(t) = Attending\_physician(u)$ means that subjects t and u have the same attending physicians,
- If t and u are terms of type *Subject*, then $Age(t) = Age(u)$ means that subjects t and u have the same age and
- If t and u are terms of type *Subject*, then $Blood\_group(t) = Blood\_group(u)$ means that subjects t and u have the same blood group.

Obviously, there are cases where other concrete binary relations must be considered. For instance:

- If t and u are terms of type *Subject*, then $Attending\_physician(t) \cap Attending\_physician(u) \neq \emptyset$ means that subjects t u have a common attending physician,
- If t and u are terms of type *Subject*, then $Age(t) < Age(u)$ means that subject t is younger than subject u and

- If t and u are terms of type $Subject$, then $Blood\_group(t) \sim Blood\_group(u)$ means that the blood groups of t and u are compatible.

Such kind of formulas will also be considered as atomic formulas. Finally the formulas of $\mathcal{L}$ are defined as follows:

- an atomic formula is a formula,
- if A is a formula then $\neg A$ "not A" is a formula,
- if A and B are formulas then $(A \wedge B)$ "A and B" and $(A \vee B)$ "A or B" are formulas and
- if A is a formula and x is an individual variable then $\forall x A$ "for all possible values of variable x, A" and $\exists x A$ "there exists possible values of variable x such that A" are formulas.

We shall use in our expressions the shorthands $\rightarrow$ and $\leftrightarrow$ as in Boolean logic: $(A \rightarrow B)$ is $(\neg A \vee B)$, and $(A \leftrightarrow B)$ is $((A \rightarrow B) \wedge (B \rightarrow A))$. We shall also omit parentheses when there is no risk of ambiguity. Examples of formulas are:

- $\forall s(Employ(Rangueil, s, Physician) \rightarrow Employ(Rangueil, s, Treating\_staff))$ "every physician in the Rangueil hospital is also employed as a treating staff" and
- $\forall r \forall v \forall a(Permission(Rangueil, r, v, a, attending\_physician) \rightarrow Permission(Rangueil, r, v, a, urgency))$ "if Rangueil hospital grants role r permission to perform activity a on view v in the context attending physician then it grants the corresponding permission within the context urgency".

The truth value of a formula is determined by the values of its subformulas in a given model. The models for our language consists, first, of 8 nonempty sets corresponding to the 8 entities of our diagram and, second, of 12 relations corresponding to the 12 relationships of our diagram. We shall assume that the precise formulation of the key definition of a formula being true in a model is already quite familiar to the reader.

Axioms of a first-order theory are usually subdivided into logical axioms and nonlogical axioms. The logical axioms provide a basis for proving all theorems of first-order classical logic, whereas the nonlogical axioms deal with some specific subject matter. Apart from the logical axioms, all security policies will be based on the following list of nonlogical axioms for all organizations org:

1. $\forall s \forall o \forall \alpha \forall r \forall v \forall a \forall c$ $Permission(org, r, v, a, c) \wedge$ $Employ(org, s, r) \wedge$ $Use(org, o, v) \wedge$ $Consider(org, \alpha, a) \wedge$ $Define(org, s, o, \alpha, c) \rightarrow Is\_permitted(s, o, \alpha)$: if organization org, within the context c, grants role r permission to perform activity a on view v, if org employs subject s in role r, if org uses object o in view v, if org considers that action α falls within the activity a and if, within org, the context c is true between s, o and α then s has permission to perform α on o,
2. $\forall r \forall v \forall a \forall c(Obligation(org, r, v, a, c) \rightarrow Recommendation(org, r, v, a, c))$: every obligation is also a recommendation,
3. $\forall r \forall v \forall a \forall c(Recommendation(org, r, v, a, c) \rightarrow Permission(org, r, v, a, c))$: every recommendation is also a permission and
4. $\forall r \forall v \forall a \forall c(Permission(org, r, v, a, c) \rightarrow \neg Prohibition(org, r, v, a, c))$: no permission is a prohibition.

Axiom 1 describes how abstract permissions between roles, views and activities can be transformed into concrete permissions between subjects, objects and actions. Similarly, axioms for obligation, recommendation and prohibition are defined.

6 Example of a security policy

In this section, we show how a simple security policy can be expressed in our first-order language.

6.1 Subjects and roles

We consider only one organization: Purpan hospital (see figure 7). Purpan hospital employs several subjects: John as a director, Mary as an administrative assistant, ST1 as a surgical team and RT2 as a radiological team. In our language, this is represented by the following instances of the $Employ$ relationship:

- $Employ(Purpan, John, director)$,
- $Employ(Purpan, Mary, administrative\_assistant)$,
- $Employ(Purpan, ST1, surgical\_team)$ and
- $Employ(Purpan, RT2, radiological\_team)$.

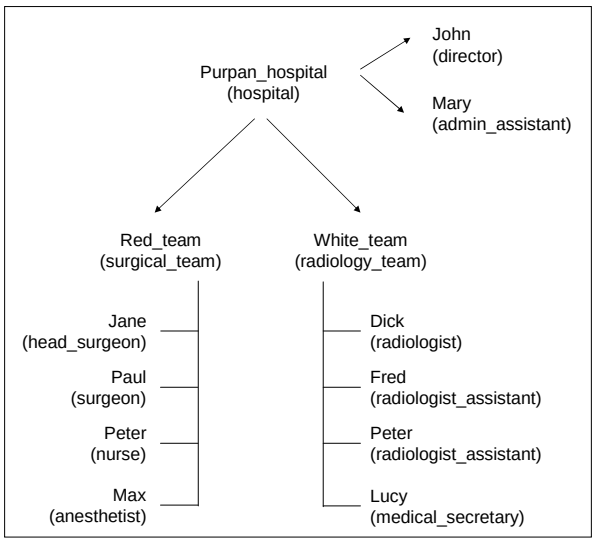

Figure 7: A simple organization

In these facts, $director$, $administrative\_assistant$, $surgical\_team$ and $radiological\_team$ are roles. We can similarly specify that the sub-organization $ST1$ employs other subjects: Jane as the head of the surgical team, Paul as a surgeon, Peter as a nurse and Max as an anaesthetist. In our language, this is represented by other instances of the relationship $Employs$ such as:

- $Employ(ST1, Jane, head\_surgeon)$,
- $Employ(ST1, Paul, surgeon)$,
- $Employ(ST1, Peter, nurse)$ and
- $Employ(ST1, Max, anaesthetist)$.

Various subjects employed by the radiological team would be similarly represented. In the sequel, we shall consider that the entity $Subject$ is associated with an attribute $Patient$ that provides the set of patients of any given subject. Therefore, our language includes a partial function $Patient$ with domain $Subject$ and range a set of finite sets of names. For instance $Patient(Purpan)$ and $Patient(Dick)$ provide the sets of patients associated to Purpan hospital and Dick respectively.

6.2 Objects and views

We shall consider objects that belong to the following views:

- $administrative\_record$: objects belonging to this view provide various administrative data about patients such as their name, their address, their age, etc,
- $medical\_record$: this corresponds to the patient medical records and
- $surgical\_record$: this corresponds to private records managed by the surgical team.

We consider that objects belonging to these views have an attribute, called $Name$. If F31.doc is a record belonging to one of these views, then $Name(F31.doc)$ provides the name of the patient who is associated with record F31.doc. We also assume that various records are directly managed by Purpan hospital, for instance in a relational database. In our model, this means that various facts have the following forms:

- $Use(Purpan, F31.doc, administrative\_record)$,
- $Use(Purpan, F32.doc, medical{-}record)$ and
- $Use(Purpan, F33.tex, surgical{-}record)$.

ST1, the surgical team, and RT2, the radiological team, actually share the database managed by Purpan hospital. This means that they use the same objects in the same view as the hospital. This can be expressed by the following rules:

- $\forall o \forall v(Use(Purpan, o, v) \rightarrow Use(ST1, o, v)$ and
- $\forall o \forall v(Use(Purpan, o, v) \rightarrow Use(RT2, o, v)$.

We can define a fourth view, called $patient\_record$ with three attributes $Administrative\_record$, $Medical\_record$ and $Surgical\_record$ as follows:

- $\forall o(Use(Purpan, o, patient\_record) \leftrightarrow$
  $\exists o_1 \exists o_2 \exists o_3 Use(Purpan, o_1, admin\_record) \wedge$
  $Use(Purpan, o_2, medical\_record) \wedge$
  $Use(Purpan, o_3, surgical\_record) \wedge$
  $Administrative\_record(o) =$
  $o_1 \wedge Medical{-}record(o) = o_2 \wedge$
  $Surgical{-}record(o) = o_3 \wedge Name(o_1) =$
  $Name(o_2) = Name(o_3))$.

The view $patient\_record$ corresponds to the complete patient's record. In a relational database, this would be obtained by joining the views $administrative\_record$, $medical\_record$ and $surgical\_record$, considered as relations, through attribute $Name$.

6.3 Actions and activities

We only consider activities that perform a direct access to the records, namely *creation*, *consulting*, *writing*, etc. If we assume that the records are managed through a relational database, these activities will respectively correspond to actions such as *insert*, *select*, *update*, etc. This is specified by rules such as:

- $Consider(Purpan, select, creation)$,
- $Consider(Purpan, select, consulting)$ and
- $Consider(Purpan, update, writing)$.

6.4 Context

We shall only model two different contexts in our example: "attending physician" and "attending team". The context "attending physician" is defined within ST1, the surgical team, as follows:

- $\forall s \forall o \forall \alpha(define(ST1, s, o, \alpha, attending\_physician) \leftrightarrow Name(o) \in Patient(s))$ that is, in ST1, the context "attending physician" is true between subject s, object o and action α if and only if o is a record corresponding to a patient of subject s.

The context "attending team" is defined as follows:

- $\forall s \forall o \forall \alpha(define(ST1, s, o, \alpha, attending\_team) \leftrightarrow \exists r(Employ(ST1, s, r) \wedge Name(o) \in patients(ST1))$ that is, in ST1, the context "attending team" is true between subject s, object o and action α if and only if s plays some role in ST1 and o is a record corresponding to one of the patients treated by ST1.

As for the context *urgency*, it may be defined as follows within ST1:

- $\forall s \forall o \forall \alpha(define(ST1, s, o, \alpha, urgency) \leftrightarrow true)$ that is, in ST1, the context "urgency" is always true between subjects, objects and actions.

6.5 Security policy

We have no room in this paper to develop a complete specification of a security policy that applies to an hospital. We shall only present few examples to illustrate the capabilities of our model. So let us consider the following permissions:

- $Permission(RT1, physician, medical\_record, consulting, attending\_physician)$,
- $Permission(RT2, physician, medical\_record, consulting, attending\_team)$ and
- $Permission(RT2, physician, surgical-record, consulting, attending\_team)$.

First permission specifies that RT1 permits physicians to consult a medical record if this medical record corresponds to a patient of these physicians. Second and third permissions specify that RT2 permits physicians to consult a medical or a surgical record if this record corresponds to a patient of RT2. As one may notice, the permissions associated with the role physician can change from one organization to another and the respective context may be also different. In our example, this is especially useful since the circumstances where a physician will consult a medical record may be different in a surgical team or a radiology team.

6.6 Hierarchies

Up to now, we do not discuss the notion of role hierarchy in our model. This notion was first introduced in the RBAC model [6] so that permissions are inherited through this hierarchy. In our approach, role hierarchy is not modelled as a basic concept. Inheritance of permissions, within ST1 between a role r_1, for instance physician, and role r_2, for instance surgeon, is specified by the following rule:

- $\forall v \forall a \forall c(Permission(ST1, r_1, v, a, c) \rightarrow Permission(ST1, r_2, v, a, c)$.

So, we may insert a relation $Sub\_role(ST1, r_1, r_2)$ in our language but instances of this relation would be simply defined as equivalent to the above rule. Notice however that, in our model, we can specify that inheritance between two given roles only applies to a given organization and would be false in another one. For instance, we can specify that, in Purpan hospital, the role director inherits the permissions of the role physician. This would be expressed by the following rule:

- $\forall v \forall va \forall c(Permission(Purpan, physician, v, a, c) \rightarrow Permission(Purpan, director, v, a, c)$.

But of course, this rule might not be available if we change *Purpan* into another hospital in which it would be possible to be director without being physician. In our model, it is also possible to specify that inheritance between roles applies to prohibition, obligation or recommendation. It is also useful to specify a hierarchy between views and to consider that permissions are inherited through this hierarchy, and similarly for a hierarchy between activities. For instance,

we may consider that views *administrative_record*, *medical_recod* and *surgical_record* are sub-views of view *patient_record*, so that a given role who is permitted to perform an activity on view *Patient_record* would also be permitted to perform the same activity on the sub-views. In our language, this would be expressed by the following rule:

- $\forall r \forall a \forall c(Permission(Purpan, r, patient\_record, a, c) \rightarrow Permission(Purpan, r, administrative\_record, a, c)$

and similarly for the views *medical_record* and *surgical_record*.

7 Constraints

Constraints were introduced in the RBAC model [6]. In our model, constraints are expressed by rules that apply to various relations. We shall only give few examples:

- $\forall s(Employ(Purpan, s, surgical\_team) \rightarrow (\exists s_1 Employ(s, s_1, surgeon) \wedge \exists s_2 Employ(s, s_2, anaesthetist) \wedge \exists s_3 Employ(s, s_3, nurse))$: this rule says that, if Purpan hospital employs s as a surgical team, then s employs a surgeon, an anaesthetist and a nurse,
- $\forall s \neg(Employ(Purpan, s, surgeon) \wedge Employ(Purpan, s, anaesthetist))$: this rule says that, in Purpan hospital, no subject s can be employed both as a surgeon and an anaesthetist and
- $\forall s \forall s'(Employ(Purpan, s, director) \wedge Employ(Purpan, s', director) \rightarrow s = s'$: this rule says, in Purpan hospital, there is only one user who is playing the role director.

We can similarly express other constraints that apply to the relationships *Use*, *Consider*, *Define*, *Permission*, *Obligation*, *Prohibition* or *Recommendation*.

8 Conclusion

This paper has presented a new security policy model that aims to solve several limits of previous models. This model, called ORBAC, is centered on the concept *Organization*. All other concepts that are used to define a security policy depend on a given organization:

- how this organization is employing subject, this is modelled through the concept *Role*,
- how this organization is using objects, this is modelled through the concept *View*,
- how this organization is performing actions, this is modelled through the concept *Activity* and
- how this organization is defining contexts that apply to users who are performing actions on objects, this is modelled through the relationship *Define*.

Using these concepts, a security policy that applies to a given organization is defined as a collection of permissions, prohibitions, obligations and recommendations. A permission corresponds to a fact having the form $Permission(org, r, v, a, c)$ to be read, in organization org, within context c, role r is permitted to perform activity a on view v. *Prohibition*, *Obligation* and *Recommendation* are similarly interpreted. We also have shown how to derive concrete permissions, prohibitions, obligations and recommendations that apply to subjects, objects and actions. Several problems have not been addressed in this paper. First, it may happen that the security policy is conflicting. For instance, for a given subject, a given object and a given action, we have to detect and solve situations where it is possible to derive both a concrete permission and a concrete prohibition. Several proposals have been suggested to deal with this problem[17, 18, 19]. The approach we suggest to solve this problem is based on possibilistic logic which enables us to automatically derive priority between facts that represent the security policy[20]. Due to space limitation, we do not address the problem of administrating a security policy defined in our model. A complete model should clearly include such an administration model. For instance, [21] suggests the ARBAC model to administrate RBAC security policy. The administration model for ORBAC will be presented in a forthcoming paper. Finally, we have also to show how to specify security properties in the ORBAC model. In particular, we have to include means to specify when the security policy is violated, and what happens in this case, for instance when an obligation is not enforced. This represents further work that remains to be done.

References

[1] B. Lampson, "Protection," in *5th Princeton Symposium on Information Sciences and Systems,*, March 1971, pp. 437–443.

[2] D. E. Bell and L. J. LaPadula, "Secure computer systems: Unified exposition and multics interpretation," Tech. Rep. ESD-TR-73-306, The MITRE Corporation, March 1976.

[3] K. J. Biba, "Integrity consideration for secure computer systems," Tech. Rep. MTR-3153, The MITRE Corporation, June 1975.

[4] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 222–274, August 2001.

[5] S. I. Gavrila and J. F. Barkley, "Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management," in *Third ACM Workshop on Role-Based Access Control*, october 22–23 1996, pp. 81–90.

[6] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C.E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[7] R. Thomas and R. Sandhu, "Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management," in *11 th IFIP Working Conference on Database Security*, Lake Tahoe, California, USA, 1997.

[8] Roshan K. Thomas, "TMAC: A primitive for Applying RBAC in collaborative environment," in *2nd ACM, Workshop on RBAC*, FairFax, Virginia, USA, November 6–7 1997, pp. 13–19.

[9] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera, "Obligation Monitoring in Policy Management," in *International Workshop, Policies for Distributed Systems and Neworks (Policy 2002)*, Monterey CA, June 5–7 2002.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," in *International Workshop, Policies for Distributed Systems and Neworks (Policy 2001)*, Bristol, UK, 2001, pp. January 29–31.

[11] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in Operating Systems," *Communication of the ACM*, vol. 19, no. 8, pp. 461–471, August 1976.

[12] J. Barkley, K. Beznosoz, and J. Uppal, "Supporting Relationships in Access Control Usiong Role Based Access Control," in *Proceeding for the ACM workshop on RBAC*, Fairfax, Virginia, USA, October 28–29 1999.

[13] E. C. Cheng, "An Object-Oriented Organizational Model to Support Dynamic Role-based Access Control in Electronic Commerce Applications," in *32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*, Maui, Hawaii, January 5–8 1999.

[14] E. Bertino, P.A. Bonatti, and E. Ferrari, "TBAC: A Temporal Role-Based Access Control for the world wide web," in *Fifth ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 2000.

[15] J.B.D. Joshi, E. Bertino, and A. Ghafoor, "Temporal Hierarchies and Inheritance Semantics for GTRBAC," in *Seventh ACM Symposium on Access Control Models and Technologies (SACMAT 02)*, Monterey, California, USA, June 2002.

[16] R. Viviani, "A Type/Domain Security Policy for Internet Trasmission Sharing and Archiving of Medical and Biological Data," in *International Workshop, Policies for Distriuted Systems and Networks (Policy 01)*, Bristol, UK, January 2001.

[17] E. Bertino, S. Jajodia, and P. Samarati, "Supporting Multiple Access Control Policies in Database Systems," in *IEEE Symposium on Security and Privacy*, Oakland, USA, 1996.

[18] G. Dinolt, L. Benzinger, and M. Yatabe, "Combining Components and Policies," in *Proc. of the Computer Security Foundations Workshop VII*, Franconia, USA, 1994.

[19] F. Cuppens, L. Cholvy, C. Saurel, and J. Carrère, "Merging Regulations: analysis of a practical example," *International Journal of Intelligent Systems*, vol. 16, no. 11, November 2001.

[20] S. Benferhat, R. El Baida, and F. Cuppens, "Modlisation des politiques de scurit dans le cadre de la thorie des possibilits," in *Rencontres Francophones de la Logique Floue et ses Applications*, Montpellier, France, October 2002.

[21] Ravi Sandhu, Bhamidipati, and Qamar Munawer, "The ARBAC97 Model for Role-Based Administration of Roles," *ACM Transactions on Information and System Security*, vol. 2, no. 1, February 1999.