Hybrid Role Hierarchy for Generalized Temporal Role Based Access Control Model

James B. D. Joshi[#], Elisa Bertino[*], Arif Ghafoor[#]

Center for Education and Research in Information Assurance and Security (CERIAS) &

[#]School of Electrical and Computer Engineering, Purdue University, USA

{*joshij, ghafoor*}*@ecn.purdue.edu,*

[*]Dipartimento di Scienze dell'Informazione, Università di Milano, Milano, Italy

bertino@dsi.unimi.it

Abstract

Generalized Temporal Role Based Access Control (GTRBAC) model that captures an exhaustive set of temporal constraint needs for access control has recently been proposed. GTRBAC's language constructs allow one to specify various temporal constraints on role, user-role assignments and role-permission assignments. In this paper, we present the notion of different types of role hierarchies based on the permission-inheritance and role activation semantics. In particular, we look at how new hierarchical relations between a pair of roles that are not directly related can be derived through other well-defined hierarchically related roles. When the different hierarchy types coexist in a role hierarchy, inferring such derived hierarchical relations between a pair of roles can be complex. The results presented here provides a basis for formally analyzing the derived inheritance and activation semantics between every pairs of roles in a hierarchy.

Keywords**: *role based, access control, temporal hierarchy*,**

1 Introduction

Role based access control (RBAC) has emerged as a promising alternative to traditional discretionary and mandatory access control (DAC and MAC) models [3, 7, 8, 9], which have some inherent limitations [10]. Several beneficial features such as policy neutrality, support for least privilege, efficient access control management, are associated with RBAC models [2, 9]. Such features make RBAC better suited for handling access control requirements of diverse organizations. Furthermore, the concept of role is associated with the notion of functional roles in an organization, and hence RBAC models provide intuitive support for expressing organizational access control policies [2].

One of the important aspects of access control is that of time constraining accesses to limit resource use. Such constraints are essential for controlling time-sensitive activities that may be present in various applications such as workflow management systems (WFMSs), where various workflow tasks, each having some timing constraints, need to be executed in some order. To address general time-based access control needs, Bertino *et al*. propose a Temporal RBAC model (TRBAC), which has been generalized recently by Joshi *et al.* [5]. The Generalized-TRBAC (GTRBAC) model [5] incorporates a set of language constructs for the specification of various temporal constraints on roles, including constraints on their activations as well as on their enabling times, user-role assignment and role-permission assignments. In particular, GTRBAC makes a clear distinction between role *enabling* and role *activation*. A role is *enabled* if a user can acquire the permissions assigned to it, but no one has done so. An *enabled* role becomes *active* when a user acquires the permissions assigned to the role in a session. An open issue in the GTRBAC model, as well as in the TRBAC model [1] is the interplay between temporal constraints and role inheritance hierarchy.

Many researchers have highlighted the importance and use of role hierarchies in RBAC models [6, 10]. A properly designed role hierarchy allows efficient specification and management of access control structures of a system. When two roles are hierarchically related, one is called the senior and the other the junior. The senior role inherits all the permissions assigned to the junior roles. The inheritance of permissions assigned to junior roles by a senior role significantly reduces assignment overhead, as the permissions need only be explicitly assigned to the junior roles.

Even though the notion of role hierarchy has been widely investigated, the implication of the presence of temporal constraints on role hierarchies has not been fully addressed in the literature. Joshi *et. al.* [4] show that there are various temporal hierarchies possible in a GTRBAC system. The ER-RBAC96 model [10] incorporates a distinction between two types of role hierarchy: *usage* hierarchy that applies *permission-usage* semantics and *activation* hierarchy that uses *role activation* semantics. Our analysis in [4] further strengthens his argument and shows that, in presence of timing constraints on various entities, the separation of the *permission-usage* and the *role-activation* semantics provides a basis for capturing various inheritance semantics of a hierarchy in presence of temporal constraints.

In this paper, we extend the work done in [4] by addressing the permission-acquisition and role activation issues when multiple hierarchy types coexist within a role hierarchy. In particular, we analyze how hierarchical relations between a pair of roles that are not directly related can be inferred from the set of well-defined hierarchically related roles. When all the hierarchy types coexist and are defined on the same set of roles, inferring such derived relations may not be simple. To deal with the coexistence of all hierarchy types in a role hierarchy, we introduce the notion of a *conditioned derived* hierarchical relation that allows one to capture more complex inheritance and activation properties of a role hierarchy. We then introduce a set of inference rules that can be used to ascertain all possible derived relations between roles in a hierarchy, and show that the set is sound and complete.

The paper is organized as follows. In section two, we briefly present the constraints of GTRBAC. In section three, we introduce the temporal GTRBAC hierarchies. In section four, we present the inference rules for derived hierarchical relations between roles and show that they are sound and complete. We discuss related work in section five and present some conclusions and future work in section six.

Portions of this work were supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS)

2 Generalized Temporal Access Control Model (GTRBAC)

The GTRBAC model [5] is an extension of the TRBAC model [1]. The model introduces the separate notions of the *enabled* and *activated* states of role, and provides constraints and event expressions associated with both these states. An enabled role indicates that a user can activate it, whereas an activated role indicates that at least one subject has activated a role in a session. The temporal constraints in GTRBAC allows the specification of the following constraints and events:

1. *Temporal constraints on role enabling/disabling*: These constraints allow one to specify the time intervals or durations during which a role is enabled, user-role assignment or a role-permission assignment is valid.
2. *Temporal constraints on user-role and role-permission assignments*: These constructs are used to express either a specific interval or a duration in which a user/permission is assigned to a role.
3. *Activation constraints*: These constraints are used to specify restrictions on a user when s/he activates a role. These constraints may specify the duration for which a user is allowed to activate a role, or can restrict the number of user allowed to simultaneously activate a particular role.
4. *Run-time events*: A set of run-time events allows an administrator to dynamically initiate GTRBAC events, or enable duration or activation constraints. Another set of run-time events allow users to make activation requests to the system.
5. *Constraint enabling expressions*: GTRBAC includes events that enable or disable duration constraints and role activation constraints. The duration constraints may be on role enablings, user-role assignments or role-permission assignments.
6. *Triggers*: Triggers allow one to express dependency among GTRBAC events as well as capture the past events and define future events based on them.

3 Temporal Role Hierarchies

Sandhu distinguishes a role hierarchy into two types: *usage* hierarchy and *activation* hierarchy [10]. By defining an *activation* hierarchy as a superset of a *usage* hierarchy. There exist scenarios where the distinction between the two is very crucial [10]. In particular, the distinction allows capturing *dynamic* SoD constraints that may exist between hierarchically related roles. Joshi *et. al*. [4], on the other hand define three hierarchy types – *permission-inheritance-only* hierarchy (*I*-hierarchy), *activation-inheritance-only* hierarchy (*A*-hierarchy) and the consolidated *inheritance-activation* hierarchy (*IA*-hierarchy) that allows both permission and activation inheritance [4]. The formal definitions of the three different hierarchy types are presented in the remainder of this section.

Table 1 reports the various predicate notations used in the formal definitions presented below. Predicates `enabled`(*r, t*), `assigned`(*u, r, t*) and `assigned`(*p, r, t*) refer to the status of roles, and user-role and role-permission assignments at time *t*. Predicate `can_activate`(*u, r, t*) implies that user *u* can activate role *r* at time *t*. It allows us to capture the fact that a user *u* may be able to activate role *r* without being explicitly assigned to it, as it is possible in a hierarchy that incorporates the *activation-inheritance* semantics. In other words, "*u can activate r*" implies that user *u* is implicitly or explicitly assigned to role *r*. It also does not rule out the possibility that some activation or SoD constraints may prevent the actual activation of *r* by *u* at time *t*. Predicate `can_acquire`(*u, p, t*) implies that "*u can acquire permission p*" at time *t* whereas the predicate `can_be_acquired`(*p, r, t*) implies that permission "*p can be acquired through role r*" at time *t*. It is important to note that `can_activate`(*u, r, t*), `can_acquire`(*u, p, t*) and `can_be_acquired`(*p, r, t*) predicates do not assume anything about the state of a role. That is, they do not say in which state role *r* is at time *t*. For example, if `can_activate`(*u, r, t*) and `enabled`(*r, t*) hold, then a user *u*'s request to activate *r* at time *t* is granted provided there are no other activation or SoD constraints prohibiting it. However, if `can_activate`(*u, r, t*) holds but not `enabled`(*r, t*), then *u*'s request to activate *r* at time *t* is denied. Thus, predicates `can_activate`(*u, r, t*), `can_acquire`(*u, p, t*) indicate possibility rather than what actually occurs.

Predicates `active`(*u, r, s, t*) and `acquires`(*u, p, s, t*) refer to what actually occurs at time instant *t*. `active`(*u, r, s, t*) indicates that role *r* is active in user *u*'s session *s* at time *t* whereas, `acquires`(*u, p, s, t*) implies that *u* acquires permission *p* at time *t* in session *s*.

Table 3.1. Various status predicates

Predicate	Meaning
`enabled`(*r, t*)	*Role r is enabled at time t*
`u_assigned`(*u, r, t*)	*User u is assigned to role r at time t*
`p_assigned`(*p, r, t*)	*Permission p is assigned to role r at time t*
`can_activate` (*u, r, t*)	*User u can activate role r at time t*
`can_acquire` (*u, p, t*)	*User u can acquire permission p at time t*
`can_be_acquired`(*p, r, t*)	*Permission p can be acquired through role r at time t*
`active`(*u, r, s, t*)	*Role r is active in user u's session s at time t*
`acquires`(*u, p, s, t*)	*User u acquires permission p in session s at time t*

The following axioms capture the key relationships among various predicates in Table 3.1 and provide a basis for identifying precisely the permission-acquisition and role-activations that are possible or that actually occur in an RBAC system.

Axioms: *For all r*∈ `Roles`, *u*∈ `Users`, *p*∈ `Permissions`, *s*∈ `Sessions`, *and time instant t* ≥ 0, *the following implications hold*:

1. `assigned`(*p, r, t*)→ `can_be_acquired`(*p, r, t*)
2. `assigned`(*u, r, t*) → `can_activate` (*u, r, t*)
3. `can_activate` (*u, r, t*) ∧ `can_be_acquired`(*p, r, t*) → `can_acquire` (*u, p, t*)

4. $\texttt{active}(u, r, s, t) \wedge \texttt{can\_be\_acquired}(p, r, t) \rightarrow \texttt{acquires}(u, p, s, t)$

Axiom (1) states that if a permission is assigned to a role, then it *can be acquired* through that role. Axiom (2) states that all users assigned to a role *can activate* that role. Axiom (3) states that if a user *u can activate* a role *r*, then all the permissions that *can be acquired* through *r can be acquired* by *u*. Thus, for the simple case where user *u* and permission *p* are assigned to *r*, the axioms indicate that *u can acquire p*. Similarly, axiom (4) states that if there is a user session in which a user *u* has activated a role *r* then *u acquires* all the permissions that *can be acquired* through role *r*.

We note that axioms (1) and (2) indicate that permission-acquisition and role-activation semantics is governed by explicit user-role and role-permission assignments. Semantically, the use of a role hierarchy is to extend the possibility of permission acquisition and role activation beyond the explicit assignments, as we shall show next. The definitions below provide the formal semantics of the time-dependent role hierarchies. The following definitions do not consider the enabling times of the hierarchically related roles, and hence are termed *unrestricted* hierarchies. The restricted forms will be introduced later.

Definition 3.1 (*Unrestricted inheritance-only hierarchy* or *I-hierarchy*): *Let x and y be roles such that* ($x \geq^t y$), *that is, x has an inheritance-only relation over y at time t. Then the following holds:*

$$\forall p, (x \geq^t y) \wedge \texttt{can\_be\_acquired}(p, y, t) \rightarrow \texttt{can\_be\_acquired}(p, x, t)$$

x is said to be a senior role of y, and conversely y is said to be a junior role of x, with respect to the inheritance-only hierarchy.

The condition characterizing the *inheritance-only* relation provides a new way of acquiring a permission through a role by using its relation with other roles. Its semantics indicates that a permission *can be acquired* through a role by direct inheritance of all the permissions of junior roles. Thus if ($x \geq^t y$), the permissions that can be acquired through *x* include all the permissions assigned to *x* (by axiom (1)) and all the permissions that *can be acquired* through role *y* (by *c*1), which in turn include all the permissions assigned to *y* as well as all the permissions that *can be acquired* through *y*'s juniors (by axiom (1) and condition *c*1). This shows that the *I*-hierarchy is transitive. Note that the axioms and condition *c*1 do not allow *u* to activate *y*. Hence, the hierarchical relation $\geq^t$ is restricted to the *permission-inheritance* semantics only.

Definition 3. 2 (*Activation hierarchy* or *A-hierarchy*): *Let x and y be roles such that* ($x \gg^t y$), *that is, x has an activation-only relation over y at time t. Then the following holds:*

$$\forall u, (x \gg^t y) \wedge \texttt{can\_activate}(u, x, t) \rightarrow \texttt{can\_activate}(u, y, t) \quad (c2)$$

x is said to be a senior role of y, and conversely y is said a junior role of x, with respect to the activation inheritance.

Here, the *activation-only* semantics introduces a new *can activate* semantics between a user and a role . Axiom (2) states that a user is able to activate a role through explicit assignment, whereas the *A*-relation allows that through relations between roles, without a need for explicit user-role assignment. Condition (*c*2) states that if user *u can activate* role *x*, and *x* has *A*-relation over *y*, then s/he *can activate* role *y* too, even if *u* is not explicitly assigned to *y*. However, note that an explicit assignment of *u* to *y* is possible but will be redundant here. The set of axioms and condition *c*2 together allow a user *u* assigned to role *x* to activate all of *y*'s juniors. However, as condition *c*1 does not apply to an *A*-hierarchy, if ($x \gg^t y$), then *u* cannot acquire *y*'s permissions by just activating *x*. Note that the $\texttt{can\_activate}(u, x, t)$ predicate makes *A*-hierarchy transitive the same way the $\texttt{can\_be\_aquired}(p, y, t)$ makes an *I*-hierarchy so.

Definition 3.3 (*General inheritance hierarchy* or *IA-hierarchy*): *Let x and y be roles such that* ($x \gtrsim^t y$), *that is, x has an general inheritance relation over y at time t. Then the following holds*

$$(x \gtrsim^t y) \rightarrow (x \geq^t y) \wedge (x \gg^t y)$$

The *IA*-hierarchy is the most common form of hierarchy and contains both *permission-inheritance* and *activation-inheritance* aspects of a hierarchy. Hence, a user can acquire permissions of roles that are junior of roles to which s/he is assigned without activating them. At the same time, s/he may activate the junior roles even though s/he is not explicitly assigned to them. Note that the definitions do not account for the enabling times of the roles that are hierarchically related.

On a given set of roles, there may be various inheritance relations. Therefore, we require that the following *consistency* property be satisfied in a role hierarchy. (r1)

Property (*Consistency of hierarchies*): *Let* $\langle f \rangle \in \{\geq^t, \gg^t, \gtrsim^t\}$ *and* $\langle f' \rangle \in \{\geq^t, \gg^t, \gtrsim^t\}/\{\langle f \rangle\}$. *Let x and y be roles such that x* $\langle f \rangle$ *y; then the condition* $\neg(y \langle f' \rangle x)$ *must hold.*

The main purpose of a hierarchical relation is the acquisition of permission of junior roles by a senior role by use of any of the three hierarchy types. The *consistency* property ensures that a senior-junior relation between two roles in one type of hierarchy is not reversed in another type of hierarchy. Due to space limitation, we do not address here other issues concerning how various hierarchies can co-exist within the same set of roles.

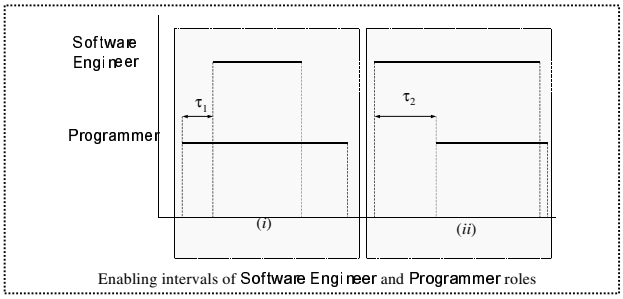

Figure 3.1 Relationship between enabling times of hierarchically related roles

The *unrestricted* hierarchies do not take into account the relationship between the enabling times of the hierarchically related roles. When we consider the enabling times of the roles,

we obtain two forms of *restricted* hierarchies – *weakly restricted* and *strongly restricted* forms. For example, Figure 3.1(*i*) illustrates a case in which there is an interval τ_1 in which the junior role is enabled but the senior role is disabled. Similarly, Figure 3.1(*ii*) illustrates a case in which there is an interval τ_2 in which the senior role is enabled and the junior role is disabled. In a *strongly-restricted* hierarchy, inheritance is not allowed during those intervals. However, in a *weakly-restricted* hierarchy, inheritance may be allowed in those intervals. Table 3.2 shows the inheritance properties of the *restricted* and *unrestricted* hierarchies in intervals τ_1 and τ_2.

Table 3.2. Inheritance semantics for the *restricted* and *unrestricted* hierarchies

r_1 is senior of r_2→		τ	τ
↓Hierarchy Type		r_1 disabled, r_2 enabled	r_1 enabled' r_2 disabled
I-hierarchy	I_w	No inheritance in τ	*Permission*-inheritance in τ
	I_s	No inheritance in τ	No inheritance in τ
A-hierarchy	A_w	*Activation*-inheritance in τ	No inheritance in τ
	A_s	No inheritance in τ	No inheritance in τ
IA-hierarchy	IA_w	*Activation*-inheritance in τ	*Permission*-inheritance in τ
	IA_s	No inheritance in τ	No inheritance in τ

4 The Inference Rules

In this section, we introduce the derived relations between two roles that may or may not be directly related. It is easy to see from the definitions of the hierarchies that each hierarchical relation has the transitive property. In a hierarchy where all three types of hierarchy can co-exit, a hierarchical relation between a pair of roles that are not directly related may be derived. We use Roles(H) to indicate the set of roles contained in a hierarchy H. A *monotype* hierarchy contains hierarchical relation of a single type applied on the roles in Roles(H), whereas, a *hybrid* hierarchy has multiple hierarchy types applied on the roles in Roles(H). While most derived relations fall into the three basic hierarchical types defined above, there exists a special derived type called a *conditioned* derived relation, written as ($x[A_1]<f>y$), where A_1 is a set of roles. A *conditioned* derived relation is defined as follows:

Definition 4.1 (*Conditioned Derived relation*): *Let H_t be a role hierarchy at time t. Let x, z, y_1, y_2,..., $y_n \in$ Roles(H_t); then $x\{y_1, y_2,..., y_n\}<f>z$ is called a* Conditioned Derived Relation (*also read as* "the derived relation $x<f>y$ is *conditioned* on a role in $\{z_1, z_2,..., z_n\}$"), *if the following conditions hold*:

$$\text{for all } y \in \{y_1, y_2,..., y_n\},\ n > 0,\ (x \succcurlyeq^t y) \wedge (y<f>z)$$

Furthermore, we write $x[Y]<f>z$ when we mean $x\{y_1, y_2,..., y_n\}<f>z$ (i.e., $[Y] = \{y_1, y_2,..., y_n\}$).

Here, the condition indicates that x is related to each $y \in \{y_1, y_2,..., y_n\}$ (directly or through a derived relation) by an A-hierarchy, whereas each y is related to z by the $<f>$ relation. This implies that a permission that can be acquired through role z can be acquired by a user u assigned to role x, without activating z, if s/he activates any of the roles in $\{y_1, y_2,..., y_n\}$. Hence, it is not required that u explicitly activate role z to acquire its permissions. Thus, while $x<f>z$ is actually the *derived relation*, an additional condition is required to be satisfied for the direct inheritance of z's permissions by users assigned to x without activating z.

As we shall see, it is not necessary that the hierarchical path from x to each y contain all A-relations; it is only required that a user assigned to or can activate x can also activate y. This implies that the hierarchical path from x to each y does not contain any I-relation, because the I-relation prohibits activation of roles below it's senior, say role x, by any user assigned to roles senior to x. Furthermore, we note that in a *conditioned derived* relation $x\{y_1, y_2,..., y_n\}<f>z$, $<f>$ is either $\succeq^t$ or $\succcurlyeq^t$ as we shall see later.

In the following, we present the inference rules for inferring all derived relations.

Definition 4.2 *(Inference Rules): Let H be a role hierarchy, x, y, z* $\in$ Roles(H), *and* $[A_1]$, $[A_2] \subseteq$ Roles(H). *Then the following inference rules are defined*:

***R*1** (*Monotype hierarchy*): $(x<f>y) \wedge (x<f>z) \rightarrow (x<f>z)$ *for all* $<f> \in \{\succeq^t, \succcurlyeq^t, \succcurlyeq^t\}$

***R*2** (*Hybrid hierarchy with unconditioned relations*):

1. $(x <f_1>y) \wedge (x <f_2> z) \rightarrow (x \succeq^t z)$ *for all* $<f_1>, <f_2> \in \{\succeq^t, \succcurlyeq^t\}$, *such that* $<f_1> \neq <f_2>$
2. $(x \succcurlyeq^t y) \wedge (x \succcurlyeq^t z) \rightarrow (x \succeq^t z)$
3. $(x \succcurlyeq^t y) \wedge (x <f> z) \rightarrow (x\{y\}<f>z)$ *for all* $<f> \in \{\succeq^t, \succcurlyeq^t, \succcurlyeq^t\}$

***R*3** (*Hybrid hierarchy with one unconditioned derived relation*):

1. $(x[A] \succeq^t y) \wedge (y <f> z) \rightarrow (x[A] \succeq^t z)$ *for all* $<f> \in \{\succeq^t, \succcurlyeq^t\}$
2. $(x[A] \succcurlyeq^t y) \wedge (y <f> z) \rightarrow (x[A] <f> z)$ *for all* $<f> \in \{\succeq^t, \succcurlyeq^t\}$
3. $(x[A] \succcurlyeq^t y) \wedge (y \succcurlyeq^t z) \rightarrow (x \succcurlyeq^t z)$

***R*4** (*Hierarchy with multiple paths between two roles*):

1. $(x<f>y)_1 \wedge (x<f> y)_2 \rightarrow (x<f>y)$ *for all* $<f> \in \{\succeq^t, \succcurlyeq^t, \succcurlyeq^t\}$ (*Monotype*)
2. $(x<f_1>y)_1 \wedge (x<f_2>y)_2 \rightarrow (x \succcurlyeq^t y)$ *for all* $<f_1>, <f_2> \in \{\succeq^t, \succcurlyeq^t, \succcurlyeq^t\}$ *such that* $<f_1> \neq <f_2>$
3. *for all* $<f>,<f_1>, <f_2> \in \{\succeq^t, \succcurlyeq^t\}$ *such that* $<f_1> \neq <f_2>$,
   a. $(x[A_1] <f>y)_1 \wedge (x<f>y)_2 \rightarrow (x<f> y)$
   b. $(x[A_1] <f>y)_1 \wedge (x \succcurlyeq^t y)_2 \rightarrow (x[A_1] <\succcurlyeq^t y)$
   c. $(x[A_1] <f_1>y)_1 \wedge (x<f_2>y)_2 \rightarrow (x \succcurlyeq^t y)$
4. *for all* $<f>,<f_1>, <f_2> \in \{\succeq^t, \succcurlyeq^t\}$ *such that* $<f_1> \neq <f_2>$, (*we have* $[A_1 \cup A_2] = [A_1] \cup [A_2]$)
   a. $(x[A_1] <f>y)_1 \wedge (x[A_2] <f>y)_2 \rightarrow (x[A_1 \cup A_2]<f> y)$
   b. $(x[A_1] <f_1>y)_1 \wedge (x[A_2] <f_2>y)_2 \rightarrow (x[A_1 \cup A_2] \succcurlyeq^t y)$

Rule ***R*1** is a trivial case of transitivity using a single hierarchy type. The transitivity follows directly from the *can be acquire* and *can activate* semantics used in the conditions *of I* and *A*-hierarchies. Thus, if <*f*> is $\succcurlyeq^t$, then from the two relations $x \succcurlyeq^t y$ and $y \succcurlyeq^t z$, relation $y \succcurlyeq^t z$ is inferred. Rule ***R*2** applies to all the cases where two different types of hierarchical relations exist in a hierarchical path. This can result in a *conditioned* derived relation. In particular, such a relation is derived whenever an *A*-relation is followed by another type. Rule ***R*3** deals with each of the cases in which an unconditioned relation (direct or derived) follows a *conditioned* derived relation. Note that the *conditioned* derived relation can be either *I*-relation or an *IA*-relation but not an *A*-relation. Hence, rules ***R*3.1** and ***R*3.2** deal only with *I*-relation and the *IA*-relation. As an *A*-relation does not allow permission inheritance, the resulting derived relation does not have any inheritance semantics (*conditioned* or *unconditioned*). In a hierarchy, there may be more than one relation between a pair of roles. Such a situation arises when there are multiple hierarchical paths between the two roles. In such a case, we need to have a clear notion of which of the derived relations between the two roles we should take. Rule ***R*4** deals with such cases. The first rule, i.e., ***R*4.1**, is a trivial case in which both the hierarchical paths result in the same *unconditioned* relation. The second rule, i.e., ***R*4.2**, takes care of all the possible combinations of two different hierarchical unconditioned relations (direct or derived) between the same pair. Similarly, the third part, i.e., ***R*4.3**, takes care of all the possible combinations of two different hierarchical relations between the same pair in which one relation is an unconditioned derived relation, whereas ***R*4.4**, takes care of all the possible combinations of two different hierarchical conditioned derived relations. In the next section, we show that the rules are sound and complete.

4.1 Soundness and Completeness of The Inference Rules

In this section, we show that the set of inference rules introduced above is sound and complete. In order to do that we use the notion of *authorization consistent hierarchies*, which is defined below. In the definition, predicate `can_activate` (*u, r, t, H*) states that *u* can activate role *r* using role activation semantics in role hierarchy *r* at time *t*. Similarly, predicate `can_be_acquired` (*p, r, t, H*) states that permission *p* can be acquired through role *r* at time *t* using permission-acquisition semantics in hierarchy *H*. UAH(*H*) is the set of all the user role assignments related to roles in Roles(*H*), whereas PAH(*H*) is the set of all role-permission assignments associated with the roles in Roles(*H*).

Definition 4.1.1 (*Authorization consistent hierarchies*): *Let* H_1 *and* H_2 *be two hierarchies over role set* `Roles`, *such that* $\mathsf{Roles}(H_1) = \mathsf{Roles}(H_2)$, $\mathsf{UAH}(H_1) = \mathsf{UAH}(H_2)$ *and* $\mathsf{PAH}(H_1) = \mathsf{PAH}(H_2)$; *then we say that* H_1 *and* H_2 *are* authorization consistent ($H_1 \approx H_2$) *if for all* $r \in \mathsf{Roles}(H_1)$, *the following conditions hold*

1. $\forall u \in$ `Users`, `can_activate` $(u, r, t, H_1) \leftrightarrow$ `can_activate` (u, r, t, H_2) , *and*
2. $\forall p \in$ `Permissions`, `can_be_acquired` $(p, r, t, H_1) \leftrightarrow$ `can_be_acquired` (p, r, t, H_2).

First, we note that, here, the two role hierarchies considered have the same role set. Furthermore, the user-role assignment and role-permission assignments associated with each role in the two hierarchies are the same. Condition (1) implies that if a user *u* can activate a role *r* in $\mathsf{Roles}(H_1)$ under hierarchy H_1, then s/he can activate it even if H_1 is replaced by H_2 (and vice versa). Similarly, the second condition says that the set of permissions that can be acquired through a role under H_1 is also the same set of permissions that can be acquired through that role in H_2. The significance of this is that if two hierarchies are *authorization consistent* then a user assigned to roles in the hierarchies can activate exactly the same set of roles and acquire the same set of permissions under the two hierarchies. As each role in the two hierarchies allows exactly the same set of permissions to be acquired through it, and the role-permission assignments in the two hierarchies are the same, it follows that, although the sets of hierarchical relations in the two hierarchies may be different, they allow the same set of permission inheritance and acquisition through each role. We use this notion of *authorization consistency* between two hierarchies to show that the set of rules presented above is sound, i.e., each new derived relation that can be deduced from a given hierarchy using the rules produces the same inheritance and activation semantics that is already present in the original hierarchy. The following theorem formally states this result.

Theorem 4.1.1 (Soundness of rules *R*1-*R*4): *Given a role hierarchy H, if a new hierarchical relation* $h = x\langle f\rangle z$ *or* $h = x[Y]\langle f\rangle z$ *is derived from hierarchical relations in H as per rules* ***R*1-*R*4,** *and* $H' = H \cup \{h\}$, *then H and H'* are authorization consistent, i.e. $H \approx H'$.

The theorem implies that a relation derived from the hierarchical relations in *H* using the rules ***R*1-*R*4** does not violate the permission inheritance and role activation conditions of *authorization consistent* hierarchies. In other words, the new derived relation does not allow a user to inherit more (or less) permissions than was allowed to him before the derived relation is added. Similarly, the new derived relation does not allow a user to be able to activate more (or less) number of roles than that was allowed before the derived relation is introduced.

Before we present the completeness theorem for the rules ***R*1-*R*4**, we introduce the following shorthand expression for a hierarchical relation that is either direct or derived. Within a hierarchy *H*, we use $h_{x,z}$ to represent ($x\langle f\rangle z$) or $x\{y_1, y_2, \ldots, y_n\}\langle f\rangle z$ for some $\langle f\rangle \in \{\geq^t, \gtrsim^t\}$, where $x, z, y_1, y_2, \ldots, y_n \in \mathsf{Roles}(H)$. Furthermore, we write $H[\mathbf{R1}\text{-}\mathbf{R4}] \models h_{x,z}$ to indicate that the relations in *H* can logically derive relation $h_{x,z}$ using rules ***R*1-*R*4**.

Lemma 4.1.1 (Completeness of rules *R*1 in monotype linear hierarchy): *Given a monotype linear hierarchy L, rule* ***R*1** *is* complete *with respect to L*; *That is*, *if for any pair of roles x, z* $\in \mathsf{Roles}(L)$

$$\neg L[\mathbf{R1}] \models h_{x,z}$$

where $h_{x,z}$ *is a derived relation, then* $L \not\approx L \cup \{h_{x,z}\}$, *i.e., the hierarchies L* and $L' = L \cup \{h_{x,z}\}$ *are not* authorization consistent.

Lemma 4.1.2 (Completeness of rules *R*1-*R*3 in mixed linear hierarchy): *Given a mixed linear hierarchy Lm, rules* ***R*1-*R*3**

are complete *with respect to L*; *That is*, *if for any pair of roles* $x, z \in \mathsf{Roles}(Lm)$

$$\neg\, Lm[\boldsymbol{R1}\text{-}\boldsymbol{R3}] \models h_{x,z},$$

where $h_{x,z}$ *is a derived relation, then* $Lm \not\approx Lm \cup \{h_{x,z}\}$, *i.e*., *the hierarchies Lm* and $Lm' = Lm \cup \{h_{x,z}\}$ *are not* authorization consistent.

Theorem 4.1.2 (Completeness of rules *R1-R4*): *Given a role hierarchy H*, *rules* ***R1-R4*** *are complete*; *That is*, *if for any pair of roles* $x, z \in \mathsf{Roles}(H)$ *such that*

$$\neg\, H[\boldsymbol{R1}\text{-}\boldsymbol{R4}] \models h_{x,z}$$

then $H \not\approx H \cup \{h_{x,z}\}$, *i.e*., *the hierarchies H* and $H' = H \cup \{h_{x,z}\}$ *are not* authorization consistent.

The theorem indicates that if a relation, say <*f*>, between any two roles, say x and z, of $\mathsf{Roles}(H)$ cannot be derived from the hierarchical relations in H, then any role hierarchy containing such a relation $(x<f>z)$ or $x\{y_1, y_2, \ldots, y_n\}<f>z)$ is not authorization consistent with H. In other words, we can take every pair of roles (x, z) of $\mathsf{Roles}(H)$ and every possible hierarchical relation between them (including conditioned derived relations) and extend H by adding it (i.e., $(x<f>z)$ or $x\{y_1, y_2, \ldots, y_n\}<f>z)$ to get H'. If we get $H = H'$, the theorem implies that the rules ***R1-R4*** is able to derive them. Hence, this shows that the rules are complete. The detail proofs of these lemmas and theorems have not included in this paper because of the length restriction. The proofs can be easily constructed by using the transitivity of the hierarchical relations and considering all the cases of the rules.

5 Related Work

Several researchers have addressed issues related to inheritance semantics in RBAC [3, 6, 7, 10]. However, none has addressed issues concerning the inheritance relation when temporal properties are introduced and when different types of hierarchical relations co-exist in a role hierarchy. We have used the separate notion of hierarchy using *permission-usage* and *role-activation* semantics similar to the one proposed by Sandhu [10] and have strengthened Sandhu's argument that the distinction between the two semantics is very crucial in [4]. Sandhu's argument is based on the fact that the simple usage semantics is inadequate for expressing desired inheritance relation when certain *dynamic* SoD constraints are used between two roles that are hierarchically related, whereas, here, we emphasized the need for such distinction to capture the inheritance semantics in presence of various temporal constraints. In [3], Giuri has proposed an activation hierarchy based on AND and OR roles. However, these AND-OR roles can be easily simulated within Sandhu's ER-RBAC96 model that uses inheritance and activation hierarchies, making Giuri's model a special case of ER-RBAC96 [10].

6 Conclusions and future work

In this paper, we formally defined and differentiated the three types of hierarchies: *inheritance-only*, *activation-only* and *general-inheritance*. We then analyzed the derived relations that can be extracted from a predefined set of hierarchical relations. In presence of multiple types of hierarchical relations, derived relations can be *conditioned*, in which case the permission inheritance semantics need to be supported by an additional condition.

We plan to extend the present work in various directions. The first direction is an extensive investigation on what the maximum or minimum set of permissions can be acquired through each role and what set of roles can be activated by a user assigned to the specific role in a hierarchy. These issues provide insight into how the principle of least privilege can be addressed in RBAC framework. This is particularly significant because the issue of principle of least privilege, although laudably considered as a virtue of RBAC systems, has not been addressed within a formal framework. We also plan to develop an SQL-like language for specifying temporal properties for roles and the various types of inheritance relations. Finally, we plan to develop a prototype of such language on top of a relational DBMS.

References

[1] E. Bertino, P. A. Bonatti, E. Ferrari. TRBAC: A Temporal Role-based Access Control Model. *ACM Transactions on Information and System Security*, 4(4), 2001 (in print).

[2] D. F. Ferraiolo, D. M. Gilbert, and N Lynch. An examination of Federal and commercial access control policy needs. In *Proceedings of NISTNCSC National Computer Security Conference*, pages 107-116, Baltimore, MD, September 20-23 1993.

[3] L. Giuri. Role-based access control: A natural approach. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*. ACM, 1997.

[4] J. B. D. Joshi, E. Bertino, A. Ghafoor. Temporal Hierarchies and Inheritance Semantics for GTRBAC. *Seventh ACM Symposium on Access Control Models and Technologies*, June 2002, pages 74-83.

[5] J. B. D. Joshi, E. Bertino, U. Latif, A. Ghafoor. Generalized Temporal Role Based Access Control Model (GTRBAC) (Part I)– Specification and Modeling. CERIAS TR 2001-47, Purdue University, USA, 2001.

[6] J. D. Moffet. Control Principles and Role Hierarchies. In *Proceedings of 3 rd ACM Workshop on Role-Based Access Control*, November 1998.

[7] M. Nyanchama and S. Osborn. The Role Graph Model and Conflict of Interest. *ACM Transactions on Information and System Security*, 2(1):3-33, 1999.

[8] S. Osborn, R. Sandhu, Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security* (TISSEC) Volume 3, Issue 2 (May 2000) Pages: 85 - 106

[9] R. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman. Role-Based Access Control Models", *IEEE Computer* 29(2): 38-47, IEEE Press, 1996

[10] R. Sandhu. Role Activation Hierarchies", In *Proceedings of 2rd ACM Workshop on Role-based Access Control*, Fairfax, Virginia, October 22-23, 1998.