# SARBAC07: A Scoped Administration Model for RBAC with Hybrid Hierarchy

Yue Zhang
*Department of Computer Science*
*University of Pittsburgh*
*Pittsburgh, PA, USA*
zysxqn@cs.pitt.edu

James B.D. Joshi
*School of Information Science*
*University of Pittsburgh*
*Pittsburgh, PA, USA*
jjoshi@sis.pitt.edu

## Abstract

*Recently, administration of RBAC systems using a role-based approach has become very appealing because of the benefits that such an approach typically brings. This approach uses RBAC itself to manage RBAC policies so that the administration functions can be decentralized and made more efficient. Existing RBAC administration models, however, fail to deal with RBAC systems with hybrid hierarchy, which has been shown to be necessary to specify fine-grained RBAC policies. In this paper, we propose a Scoped Administration model for RBAC with Hybrid Hierarchy (SARBAC07) by using the notion of an administrative scope that was earlier proposed in the SARBAC model. We show that our model keeps all the advantages of the original model and can deal with more complex situations where hybrid hierarchy is needed.*

## 1. Introduction

Role Based Access Control (RBAC) has become widely accepted as a promising alternative to the traditional discretionary access control (DAC) and mandatory access control (MAC) approaches [3, 4, 5, 12]. In RBAC, permissions are assigned to roles and users are made members of roles. RBAC model is policy-neutral and flexible. Users can be easily reassigned from one role to another whenever needed, and roles can also be granted new permissions or existing permissions can be easily reassigned whenever the function of a role changes.

To support evolution of RBAC policies, efficient administration of RBAC is a crucial challenge. In modern large enterprise-wide systems, there could be many roles and many more users/permissions [14]. The relationships among the roles, users, and permissions change continuously. Centralized management of such large number of roles, users, permissions and their interrelationships can have several drawbacks [14]. Hence, decentralizing the administration of RBAC without losing the central control is a challenging goal for system designers and developers.

The use of role itself to manage the RBAC policies has become an appealing idea. Sandhu *et al.* [14] have proposed an ARBAC97 (Administrate RBAC '97) model consisting of URA97 (User-Role Assignment '97), PRA97 (Permission-Role Assignment '97), and RRA97 (Role-Role Assignment '97) model, which use RBAC to manage RBAC policies. They have further extended this model to ARBAC99 [15] and ARBAC02 [11]. Crampton *et al.* have developed a Scoped Administration model for RBAC (SARBAC) model using the concept of *administrative scope* [1] to address some shortcomings of the ARBAC97 model and has been shown to be better in terms of completeness, simplicity, practicality and versatility.

However, neither of these approaches deals with RBAC policies with hybrid hierarchies – where different types of hierarchical relationship among roles can co-exist. Issues related to hybrid hierarchies have been first formally addressed by Joshi *et al.* [10]. Several researchers [9, 10, 13] have found that hybrid hierarchy is necessary when more fine-grained RBAC policies are needed, in particular, when we need to specify dynamic separation of duty (DSoD), temporal and cardinality constraints on roles in a hierarchy. Joshi *et al.* have introduced three types of hierarchy relations by separating the permission inheritance semantics (in *I*-hierarchy type) and activation inheritance semantics (in *A*-hierarchy type). Roles related by an *A*-hierarchy can be constrained by a DSoD constraint [6]. Joshi *et al.* also show that *A*-hierarchy is suitable for *permission-centric* cardinality constraints, while *I*-hierarchy or *IA*-hierarchy (which allows both permission and activation inheritance) is suitable for *user-centric* cardinality constraints. Further more, Du *et al.* [2] show that hybrid hierarchy is particularly useful when we want to map the policies in multi-domain applications.

In this paper, we redefine the concept of *administrative scope* to develop a *Scoped Administration model for RBAC with Hybrid Hierarchy* (SARBAC07) to administer RBAC systems that support hybrid hierarchies. We also show that the User-Role Assignment and Permission-Role-Assignment operations defined in the SARBAC model have some ambiguity because of the use of the role hierarchy proposed in the NIST's RBAC [4] (We refer to this as "standard hierarchy" in this paper; also note that it is same as the *IA*-hierarchy type). We show that we are able to solve this ambiguity by using our proposed model. In summary, this paper has two major contributions:

0-7695-2876-7/07 $25.00 © 2007 IEEE
DOI 10.1109/IAS.2007.90

1. *We propose a SARBAC07 model which can deal with RBAC policies with hybrid hierarchy by redefining concepts and operations of the SARBAC model.*
2. *We solve an ambiguity in the SARBAC model by using our SARBAC07 model and show that the User-Role Assignment is determined by IA -relation while Role-Permission Assignment is determined by I-relation in hybrid hierarchy.*

The rest of the paper is organized as follows. In Section 2 we review the relevant background such as hybrid hierarchy and the SARBAC model. We propose and evaluate our SARBAC07 model in Section 3 and Section 4, and finally conclude our work in Section 6.

## 2. Background

### 2.1. Hybrid Hierarchy

Hybrid hierarchy was introduced in the context of the Generalized Time based RBAC (GTRBAC) model to facilitate specifications of fine grained RBAC policies [7]. In a hybrid hierarchy, the following three hierarchical relations among roles can co-exist: *permission-inheritance-only* hierarchy (*I*-hierarchy represented as $\geq_i$), *role-activation-only* hierarchy (*A*-hierarchy represented as $\geq_a$) and the combined *permission-inheritance-activation* hierarchy (*IA*-hierarchy represented as $\geq$) [8]. Semantically, $s \geq_i j$ means permissions available through $j$ are also available through $s$; $s \geq_a j$ means that any user who can activate $s$ can also activate $j$; and $s \geq j$ means that $s$ inherits permissions of $j$ and the users that can activate $s$ can also activate $j$. Figure 1 shows a sample hybrid hierarchy. Note that in the standard hierarchy we also use the symbol $x \geq y$ to represent the hierarchy relations.

Joshi *et al.* have shown that in a hybrid hierarchy the hierarchical relation between any pair of roles which are not directly related can be derived [8]. It is obvious that the three hierarchy types are transitive. For instance, if ($x \geq y$) and ($y \geq z$) then it implies ($x \geq z$). Similarly, since *IA*-relation can be considered as both *I*-relation and *A*-relation, we have the following relations as shown in Figure 3(a): $(x <f_1> y) \wedge (y <f_2> z) \rightarrow (x <f> z)$, *where,* $(<f_1> \in \{\geq\}) \vee (<f_2> \in \{\geq\})$ *and* $<f> = <f_1>$, *if* $<f_2> \in \{\geq\}$, *otherwise* $<f> = <f_2>$.

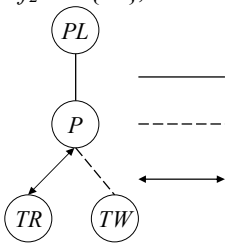
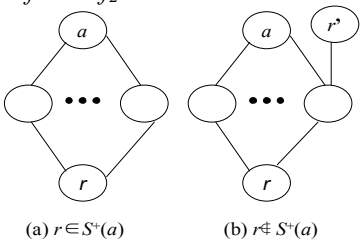



**Figure 1.** A sample hybrid hierarchy




(a) $r \in S^+(a)$ (b) $r \notin S^+(a)$

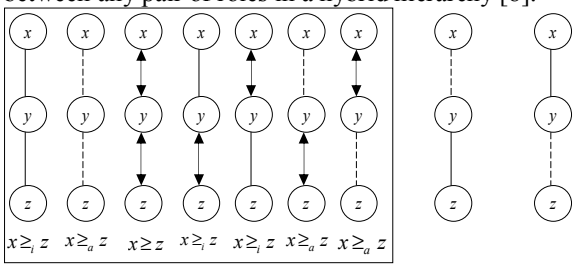
**Figure 2.** Administrative scope in SARBAC

A special case of derived relation is when an *A*-relation is followed by an *I*-relation, as shown in Figure 3(b); in this case, we should be very careful when analyzing its semantic. Here, by activating $x$, a user assigned to $x$ can not acquire the permissions of $z$, although he can acquire the permissions of $z$ by activating $y$. This means $x$ can still "inherit" permissions of $z$ even if there is no *I*-relation derived between them. In this situation, we say that $x$ has a "*conditioned*" relation with $z$, written as $x[y] \geq_i z$. In [8], the *conditioned derived* relation is defined as $x[A](B) \geq_i y$, where $B$ indicates a set of $A$-paths from $x$ to $y$. In this paper, we ignore set $B$; if $B$ is not empty, we simply consider it as $x \geq_a y$ without affecting any semantics.

Now consider the case where an *I*-relation is followed by an *A*-relation, as shown in Figure 3(c). Here, a user assigned to $x$ can not acquire the permissions of $z$, since he can only acquire the permissions of $y$ (by activating $x$) but can not activate $y$. Therefore, there's no relation between $x$ and $z$. We define the derived relations as follows:

**DEFINITION 2.4** ***(Derived Relation):*** *Let x and y be roles such that* ($x \geq_d y$)*, that is, x has a derived relation with y. Then the following holds:* $(x \geq_i y) \vee (x \geq_a y) \vee (x \geq y) \vee (\exists a \in R, x[a] \geq_i y)$

Joshi *et al.* propose a complete and sound set of inference rules to find all the possible derived relations between any pair of roles in a hybrid hierarchy [8].




(a) Unconditioned relations (b) $x[y] \geq_i z$ (c) No relation

**Figure 3.** Derived relations in a hybrid hierarchy

### 2.2. Overview of the SARBAC model.

The basic idea of the SARBAC model is to use some roles to "administer" some other roles [1]. In this way, the administration can be decentralized. The notion of *administrative scope*, as defined below, is used to define which role can administer which roles.

***DEFINITION* 2.5 (*Administrative scope*):** *Given a role a, its administrative scope,* $S(a)$*, is defined as:*

$$S(a) = \{r \in R: r \leq a\ ,\ \uparrow r \setminus \uparrow a \subseteq \downarrow a\}$$

*where,* $\uparrow r = \{x \in R: x \geq r\}$, $\downarrow r = \{x \in R: x \leq r\}$.

Informally, $r \in S(a)$ if every path upwards from $r$ goes through $a$. That is, any change to $r$ made by $a$ will not have unexpected side effects due to inheritance elsewhere in the hierarchy. The *strict* administrative scope of $r$ is defined as $S(r) \setminus \{r\}$, denoted as $S^*(r)$. If $r \in S^+(a)$, we call $a$ as an *administrator* of $r$ *[1]*. The SARBAC model has three parts: the *Role Hierarchy Administration* (RHA) model, the *User Role Assignment* (URA)

model, and the *Permission Role Assignment* (PRA) model. SARBAC-RHA defines four administration operations: *AddRole*(*a, r,* $\triangle r$, $\triangledown r$), *DeleteRole*(*a, r*), *AddEdge*(*a, c, p*), and *DeleteEdge(a, c, p)*, where $\triangle r$ and $\triangledown r$ are sets of the immediate juniors and immediate seniors of *r*, respectively Table 2 describes the conditions that needs to be satisfied for these operations to succeed. SARBAC defines a family of four RHA models, namely $RHA_1$, $RHA_2$, $RHA_3$, and $RHA_4$. The key difference among them is that $RHA_3$ and $RHA_4$ create a set of special administration roles and assign to each of them some "normal" roles to adminster. Each administration role can manage the "normal" roles assigned to it, as well as all the roles within the administrative scopes of these "normal" roles. In SARBAC-URA, operations and their success conditions are summarized in Table 3, where $\wedge C$ is a set of constraints needed to be satisfied by the users or permissions and *ua-constraints* assign some constraints to each of the role *r*. For example, the first row of Table 3 shows that if role *a* wants to assign user *u* to role *r*, *r* must be within the administrative scope of *a*, and *u* must satisfy the "pre-condition" associated with *r*. SARBAC-PRA is very similar to SARBAC-URA – with users substituted by permissions. In sub-section 3.3 we show some ambiguities associated with both the SARBAC-URA and SARBAC-PRA models.

**Table 2.** Hierarchy operations in SARBAC-RHA

| *Operation* | *Conditions* |
|---|---|
| *AddRole* (*a, r,* $\triangle r$, $\triangledown r$) | $\triangle r \subseteq S^+(a)$, $\triangledown r \subseteq S(a)$ |
| *DeleteRole* (*a, r*) | $r \in S^-(a)$ |
| *AddEdge* (*a, c, p*) | $c, p \in S(a)$ |
| *DeleteEdge* (*a, c, p*) | $c, p \in S(a)$ |

**Table 3.** User-Role operations in SARBAC-URA

| *Operation* | *Conditions* |
|---|---|
| *AssignUser(a, u, r)* | $r \in S(a)$, *u satisfies* $\wedge C$, $(r, \wedge C) \in$ *ua-constraints* |
| *RevokeUser(a, u, r)* | $r \in S(a)$ |

# 3. The SARBAC07 Model

## 3.1. Administrative Scope in SARBAC07

As discussed earlier, a role *r* can be administered under another role *a* if and only if all path upwards from *r* go through *a*, as shown in Figure 2(a). On the contrary, suppose there is a path upwards from *r* that doesn't go through *a*, and instead, goes through role *r'*, as shown in Figure 2(b). Here *a* and *r'* have no relation between them, but both of them are related to *r*. If *a* makes some changes to *r*, then it would also affect *r'*. So *a* should not be allowed to administer *r*. Note that in a standard hierarchy, if there's a "path" between two different roles $r_1$ and $r_2$, then $r_1$ and $r_2$ must be hierarchically related, i.e. $r_1 \geq r_2$ or $r_2 \geq r_1$. Therefore, the definition of administrative scope closely relies on finding the direct and indirect relations in the path between $r_1$ and $r_2$. Based on the definition of the derived relation $\geq_d$ earlier, we re-define the administrative scope as follows:

***DEFINITION* 3.1 (Administrative Scope in Hybrid Hierarchy)***: The administrative scope for role a in a hybrid hierarchy,* $S_{HH}(a)$*, is defined as follows:*

$$S_{HH}(a) = \{r \in R: r \leq_d a\ ,\ \uparrow r \setminus \uparrow a \subseteq \downarrow a\}$$

*Where,* $\uparrow r = \{x \in R: x \geq_d r\}$, $\downarrow r = \{x \in R: x \leq_d r\}$.

Similarly, the *strict* administrative scope is $S_{HH}^+(r) = S_{HH}(r) / \{r\}$. If $r \in S_{HH}^+(a)$, we call *a* as an *administrator* of *r*. Figure 4 illustrates the difference between the original administrative scope in SARBAC and the administrative scope in SARBAC07. Note that the structure of the three hierarchies is exactly the same and the only difference is the types of the hierarchical relations used. Figure 4(a) is a standard hierarchy; Figures 4(b) and 4(c) are hybrid hierarchies. In Figure 4(a), role *a* can not administer role *r* because *r'* is senior to *r* but is not junior to *a*. In figure 4 (b), role *a* can not administer role *r* either, since *r'* is "conditionally" senior to *r* but is not junior to *a*. In figure 4(c), however, role *a* can administer role *r* because there's no derived relation between *r* and *r'* even if there seems to be a "path" between them. Note that in Figure 4(c), *a* can not administer $r_1$ because of *r'*. However, in the entire hierarchy, there may exist another role (e.g., the senior role of both *a* and *r'*) which can administer $r_1$. Next, we will show that our definition of administrative scope provides better flexibility and maintains the decentralization/autonomy properties.

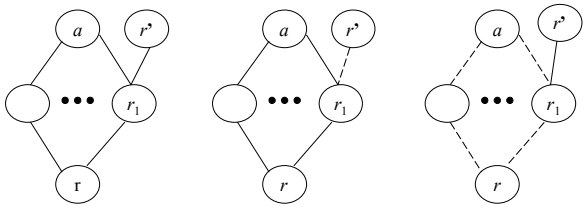



**Figure 4.** Administrative Scope in SARBAC and SARBAC07

**Flexibility:** The administrative scope in both the models is determined by the role hierarchy itself, and it changes dynamically as the hierarchical relations change. The semantics of the hierarchy type affects the different scenarios in our model. This also provides more fine-grained semantics, and hence more flexibility.

**Decentralization and Autonomy:** we illustrate this by proving the following proposition. We retain the notion of the *line manager* from the SARBAC model:

**PROPOSITION 3.2 (*Line Manager in Hybrid Hierarchy*)***:*** *In a hybrid hierarchy, if r has an administrator then the set of administrators of r has a unique minimal administrator, referred to as the line manager of r.*

**PROOF***: If r has a single administrator, the result follows immediately. Otherwise, suppose x and y are minimal administrators of r, i.e., for all administrators z of r,* $z \leq_d$

*x implies z = x, and z ≤$_d$ y implies z = y; hence, x ⊀ y and y ⊀ x. Then, r∈$S_{HH}^+$(x) and hence x∈↑r. Similarly, r∈$S_{HH}^+$(y) and hence ↑r∩↑y⊆↓y. x∉↑y gives x∈↓y. Thus, x<y, which is a contradiction.* ■

The line manager can serve as a "local" administrator. This provides decentralization and autonomy in administration of hybrid hierarchies.

## 3.2. RHA in SARBAC07

In addition to the four operations defined in SARBAC as shown in Table 2, we further add two operations in SARBAC07: *PartitionRole*() and *ChangeEdge*(), which are necessary for administering hybrid hierarchies. The success conditions of each operation are shown in Table 4, where $\triangle_a r$ is a set of immediate *A*-juniors of the role *r*, $\triangledown_a r$ is the set of immediate *A*-seniors of role *r*, $\triangle_i r$ is the set of immediate *I*-juniors of role *r*, and $\triangledown_i r$ is the set of immediate *I*-seniors of role r, as shown in Figure 5. The semantics of *ChangeEdge*(*a, c, p*) is straight forward since there are three types of edges in a hybrid hierarchy. In fact, we can use *AddEdge*() and *DeleteEdge*() operations to perform *ChangeEdge*(). That is, first delete the old edge, and then add the edge with the new type. The semantics of *PartitionRole*() is complex. Specifically, we can partition a given role vertically, horizontally, or both [8].

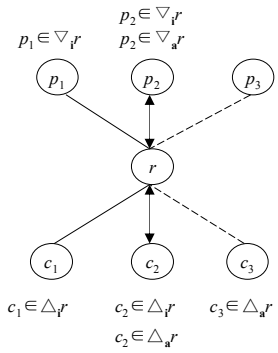



**Figure 5.** Parameters in AddRole

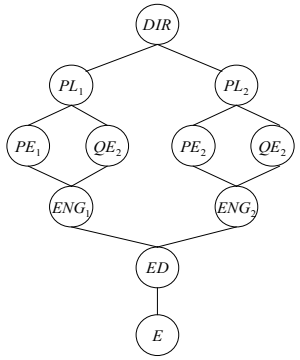



**Figure 7.** A standard hierarchy

We need to maintain the administrative scope during those operations by satisfying the following conditions:

$C_1$: *After AddRole*() *and PartitionRole*() *operations, the new role(s) should be within a's administrative scope.*

$C_2$: *After each operation, the original roles' administrators should not be changed.*

It is obvious that $C_1$ is satisfied according to our definition. Since all the seniors of the new role should be administered by *a*, the new role itself is also administered by *a*. $C_2$ is also satisfied for all operations. This conclusion is not obvious with *ChangeEdge*() operation, since the operation itself may change the relation between roles and thus affect the administrative scope, as shown in Figure 6. In Figure 6(a), $r \in S_{HH}(a)$. If we change the edge ($r$, $r_1$) to the *I*-type, as Figure 6 (b) shows, $r \notin S_{HH}(a)$ now. However, in Figure 6(a), $r_1$ is not administered by *a*, so the *ChangeEdge*() operation fails. Therefore, if *ChangeEdge*() operation succeeds, it is guaranteed that it will not affect the administrators of all the original roles.

**Table 4.** Hierarchy operations in SARBAC07

| Operation | Success Conditions |
|---|---|
| *AddRole*(*a*, *r*, $\triangle_a r$, $\triangledown_a r$, $\triangle_i r$, $\triangledown_i r$) | $\triangle_a r \subseteq S_{HH}^+(a)$ $\triangledown_a r \subseteq S_{HH}(a)$ $\triangle_i r \subseteq S_{HH}^+(a)$ $\triangledown_i r \subseteq S_{HH}(a)$ |
| *DeleteRole*(*a*, *r*) | $r \in S_{HH}^+(a)$ |
| *PartitionRole*(*a*, *r*) | $r \in S_{HH}^+(a)$ |
| *AddEdge*(*a*, *c*, *p*, *type*) | $c, p \in S_{HH}(a)$ |
| *DeleteEdge*(*a*, *c*, *p*) | $c, p \in S_{HH}(a)$ |
| *ChangeEdge*(*a*, *c*, *p*, *type*) | $c, p \in S_{HH}(a)$ |

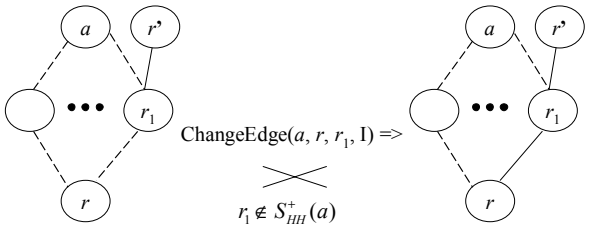



**Figure 6.** The ChangeEdge operation won't succeed

## 3.3. URA and PRA in Hybrid Hierarchy

The key operations in SARBAC-URA are shown in Table 3, and the permission-role assignment operations in SARBAC-PRA are similar. We first show that there is an ambiguity in the semantics of URA and PRA in the original SARBAC, which our model solves by redefining those operations. To illustrate these, we first review an important concept in SARBAC, the SARBAC constraint, as follows: let $R' = \{r_1, \ldots, r_k\}$ be a subset of $R$ and let $\wedge R'$ denote $r_1 \wedge \ldots \wedge r_k$.

***DEFINITION 3.3 (SARBAC constraint)*** *A SARBAC constraint has the form $\wedge C$, where $C \subseteq R$. A SARBAC constraint $\wedge C$ is satisfied by a user u if $C \subseteq \downarrow R(u)$. A SARBAC constraint $\wedge C$ is satisfied by a permission p if $C \subseteq \uparrow R(p)$, where for any $Y \subseteq X$, $\uparrow Y = \{x \in X: \exists y \in Y$ such that $x \geq y\}$, and $\downarrow Y = \{x \in X: \exists y \in Y$ such that $x \leq y\}$.*

Let's first analyze under what situation a user will satisfy a constraint. A sample standard hierarchy is shown in Figure 7, which is borrowed from [1]. According to Definition 3.3, the constraint $PE_1 \wedge QE_1$ is satisfied by any user assigned to both $PE_1$ and $QE_1$, and by any user assigned to either $PL_1$ or *DIR*. The semantics here is that any user assigned to either $PL_1$ or *DIR* is also a member of $PE_1$ and $QE_1$, and hence the $PE_1 \wedge QE_1$ constraint is satisfied. Obviously, the authors of SARBAC implicitly assumes the hierarchy relation in any standard hierarchy as "Is-a" relation [10], i.e., $x \geq y$ means any user assigned to $x$ is also a member of $y$. For example, the

leader of a team is also a member of the team. However, the semantics of standard hierarchy has long been argued as ambiguous [9, 10, 13]. The hierarchical relation in a standard hierarchy could be "Is-a", "Supervision", or "Activation" [10]. The use of hybrid hierarchy can solve this ambiguity accordingly by including three types of hierarchical relations. The above "Is-a" relation is essentially "*IA*" relation in the hybrid hierarchy, since *x* "is" *y* means any user assigned to *x* should be able to acquire all the permissions assigned to *y* through *x*, and should also be able to activate *y*. Because whether a user satisfies a constraint depends on the definition of $\downarrow Y$ in Definition 3.3, we re-define it as:

$$\forall Y \subseteq X, \downarrow Y = \{x \in X: \exists y \in Y \text{ such that } x \leq y\} \quad (1)$$

Note the symbol $\leq$ clearly means the *IA*-relation in hybrid hierarchy. Next let's analyze in what situation a permission will satisfy a constraint. In Figure 7, the constraint $PE_1 \wedge QE_1$ is satisfied by any permission assigned to both $PE_1$ and $QE_1$, and by any permission assigned to either $ENG_1$ or *ED* or *E*. The semantics here is that any permission assigned to $ENG_1$ or *ED* or *E* is also in the permission set of $PE_1$ and $QE_1$, the $PE_1 \wedge QE_1$ constraint is satisfied. In other words, $x \geq y$ means $P(y) \subseteq P(x)$, where $P(r)$ is the permission set available through *r*. Obviously, the author of SARBAC implicitly assumes the hierarchy relation in any standard hierarchy as "Permission Inheritance" relation, which is in conflict with previous assumption of "Is-a" relation. We believe this ambiguity comes from the ambiguity of the standard hierarchy, as pointed to by many researchers [9, 10, 13]. Again, the use of hybrid hierarchy can solve this by using "*I*-relation". Specifically, since a permission satisfying a constraint depends on the definition of $\uparrow Y$ in Definition 3.3, we re-define it as:

$$\forall Y \subseteq X, \uparrow Y = \{x \in X: \exists y \in Y \text{ such that } x \geq_i y\} \quad (2)$$

Note that here we use the $\geq_i$ relation. Given the new definition of $\downarrow Y$ and $\uparrow Y$, we can define the SARBAC07 constraint as follows:

***Definition 3.4 (SARBAC07 constraint):*** *A SARBAC07 constraint has the form* $\wedge C$ *for some* $C \subseteq R$. *A SARBAC07 constraint* $\wedge C$ *is satisfied by a user u if* $C \subseteq \downarrow R(u)$. *A SARBAC07 constraint* $\wedge C$ *is satisfied by a permission p if* $C \subseteq \uparrow R(p)$, *where the symbol* $\uparrow$ *and* $\downarrow$ *are defined by* (1) *and* (2).

The definition implies that the User-Role Assignment is determined by the *IA*-relation while the Permission-Role Assignment is determined by the *I*-relation. The user-role assignment operations are the same with SARBAC, as shown in Table 3 (permission-role assignment operations are similar).

## 4. Model Evaluation

In this section, we use two examples to show that our SARBAC07 model is better in terms of practicality and versatility. Also note that the SARBAC model is inadequate for the hierarchies in the examples.

**Example 4.1**: *Consider the hierarchy in Figure 1 and the following requirements for a programming project. A software tool is used for the programming task. The project leader* (*PL*) *mainly supervises the programming tasks. Only the programmers* (*P*) *do the coding. PL can only examine the tasks the P has carried out. Figure 1 depicts the hierarchy that can be generated for achieving the goal. Role TaskR* (*TR*) *contains the read-only permissions whereas role TaskW* (*TW*) *contains all the write/modify permissions related to the programming task. The role PL is I-senior to the role P. Note that users assigned to the PL can acquire permissions of TR but not of TW.*

In this example, standard hierarchy is inadequate. Since the *PL* only has the read permission of the code but can not edit the code, we can use two separate roles such as *TR* and *TW*. If we use the standard hierarchy, we would have $PL \succ TW$. However, $PL \geq P$ and $P \geq TW$ would mean $PL \geq TW$, which is in conflict with $PL \succ TW$. Therefore, we must use hybrid hierarchy (Figure 1) to satisfy all the semantics.

According to our definition, $S_{HH}(PL) = \{PL, P, TR\}$, and $S_{HH}(P) = \{P, TR, TW\}$, i.e., *PL* can not administer *TW*, only *P* can administrate *TW*, and both *PL* and *P* can administer *TR*. This is exactly the original semantics of the example 4.1. And suppose *PL* wants to change the edge (*P, TW*) to an *I-edge* so that he can also inherit the permissions of *TW*, the operation will not succeed as $TW \notin S_{HH}(PL)$. We can see that our example works well in the presence of hybrid hierarchy. To show the versatility of our model, we apply our model to a totally different scenario described in example 4.2.

**Example 4.2:** *Assume domain* 1 *and domain* 2 *both require services (a set of permissions) from each other. In a RBAC system, domain* 1 *needs to export some roles that have the set of permissions required by domain* 2, *and domain* 2 *needs to export some roles that have the set of permissions required by domain* 1. *In addition, to use the permissions of domain* 1, *domain* 2 *must create some roles through which it can access the permissions in domain* 1, *and domain* 1 *also needs to create some roles through which it can access the permissions in domain* 2. *Figure 8 shows the entire example.*

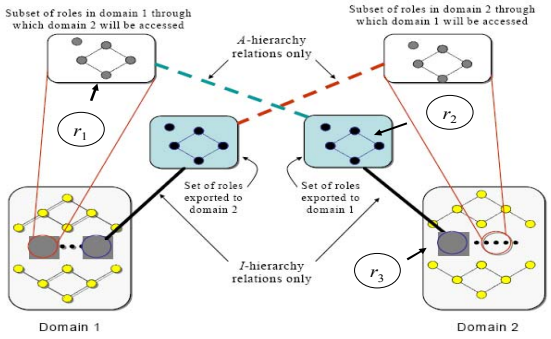



**Figure 8.** Inter-domain role mapping using hybrid hierarchy

In this example, the standard hierarchy will not work. We should use *I*-relations and *A*-relation as in Figure 9 to prevent the transitivity of the activation semantics which is usually the key problem in inter-domain access [2].

In figure 8, although $r_1$ can have the permission assigned to $r_3$ by activating $r_2$, $r_1$ can not administer $r_3$. This is because in domain 2, $r_3$ may have other seniors that have no relationship with $r_1$. However, $r_1$ can administer $r_2$ according to the definition. This is quite reasonable since $r_2$ is "exported" from domain 2 to be used by $r_1$, but $r_3$ is the "local" role in domain 2. In this way, the overall effect of our model is that roles in domain 1 can only administer the roles "specially exported" from domain 2 but can not administer the "local" roles in domain 2.

## 5. Conclusion and Future Work

In this paper, we have proposed the SARBAC07 model that can be used to administer RBAC system with hybrid hierarchies. Our model uses the key notion of administrative scope from Crampton *et al.*'s SARBAC model and redefines. We also redefine all the necessary operations accordingly. Moreover, we show that the original SARBAC model has ambiguous semantics in its User-Role Assignment and Role-Permission Assignment components, which we remove in our proposed model. Finally, we evaluate our model according to the criteria used to evaluate the SARBAC model. We plan to extend this work to construct a complete administration model for GTRBAC systems with hybrid hierarchy and constraints.

**Acknowledgement:** This research has been supported by the US National Science Foundation award IIS-0545912.

## References

[1] J. Crampton, G. Loizou, "Administrative scope: A foundation for role-based administrative models", *ACM Transactions on Information and System Security (TISSEC)*, Volume 6, Issue 2, May. 2003, pp. 201-231.

[2] S. Du, and J. B. D. Joshi, "Supporting Authorization Query and Inter-domain Role Mapping in Presence of Hybrid Role Hierarchy," The 11th ACM Symposium on Access Control Models and Technologies, USA, June 2006.

[3] D. Ferraioio, J. Cugini, and R. Kuhn, "Role-based access control (RBAC): Features and motivations", In Proceedings of 11th Annual Computer Security Application Conference, New Orleans, LA, Dec. 1995, pp. 241-48.

[4] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, " Proposed NIST standard for role-based access control," *ACM Transactions on Information and Systems Security*, vol. 4, no. 3, pp. 224–274, August 2001.

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

[6] J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Temporal hierarchies and inheritance semantics for GTRBAC", In Proceedings of the 7th ACM symposium on Access control models and technologies, New York, NY, USA, pp. 74–83.

[7] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model," *IEEE Transactions on Knowledge and Data Engineering*, Volume 7, Issue 1, Jan. 2005.

[8] J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Formal Foundations for Hybrid Role Hierarchy", *ACM Transaction in Information and Systems Security* (accepted).

[9] N. Li, "A Critique of the ANSI Standard on Role Based Access Control", to appear in *IEEE Security and Privacy*.

[10] J. D. Moffett and E. C. Lupu, "The uses of role hierarchies in access control", Proceedings of the fourth ACM workshop on Role-based access control, 1999, pp. 153-160.

[11] S. Oh, R. Sandhu, "A model for role administration using organization structure", Proceedings of the 7th ACM symposium on Access control models and technologies, Monterey, CA 2002.

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

[13] R. Sandhu, "Role activation hierarchies", Proceedings of the third ACM workshop on Role-based access control, Fairfax, Virginia, United States, 1998, pp. 33-40.

[14] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 Model for Role-Based Administration of Roles", *ACM Transactions on Information and System Security (TISSEC)*, Volume 2, Issue 1, Feb. 1999, pp. 105-135.

[15] R. Sandhu and Q. Munawer, "The ARBAC99 Model for Administration of Roles (1999)", In Proceedings of 15th Computer Security Applications Conference, Arizona, US, Feb 1999, pp. 229-2