

# SARBAC-HH: A Scoped Administration Model for RBAC with Hybrid Hierarchy

Yue Zhang, and James B.D. Joshi

<sup>1</sup>University of Pittsburgh, School of Information Science,  
735 N. Bellefield Ave., Pittsburgh PA 15260, U.S.A  
yuz20@pitt.edu jjoshi@sis.pitt.edu

**Abstract:** Recently, administration of RBAC systems using a role-based approach has become well recognized as very appealing because of the benefits that a role-based 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 an RBAC system 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 (SARBAC-HH) by using the notion of 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 requirements where hybrid hierarchy is needed.

**Keywords:** RBAC, Administration, Hybrid Hierarchy, Administrative Scope.

## 1. Introduction

Role Based Access Control (RBAC) has become widely accepted as a promising alternative to the traditional discretionary and mandatory access control (DAC and 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 if needed, and roles can also be granted new permissions or existing permissions can be easily reassigned if the function of a role changes. Significant efforts have been expended towards fine-grained RBAC policies for emerging applications.

A crucial challenge faced by emerging large application environments that employ RBAC policies is that of employing an efficient administration mechanism for RBAC policies that evolve continuously. In such 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 administer an RBAC policy has become an appealing idea. In [14], Sandhu *et al.* propose an ARBAC97 (Administrative RBAC '97) model consisting of three components that use RBAC to manage RBAC policies,

namely, the URA97 (User-Role Assignment '97) model, the PRA97 (Permission-Role Assignment '97) model, and the RRA97 (Role-Role Assignment '97) model. The ARBAC97 model has been further extended to ARBAC99 [15] and ARBAC02 [11]. Crampton *et al.* have proposed an SARBAC (Scoped Administration model for RBAC) model using the concept of *administrative scope* [1] to address some shortcomings of the ARBAC model. SARBAC has been shown to be better in terms of completeness, simplicity, practicality and versatility.

Neither ARBAC nor SARBAC deals with extended RBAC models that support hybrid hierarchies, where different types of hierarchical relationship among roles can co-exist. Issues related to hybrid hierarchies have been formally addressed by Joshi *et al.* in [10]. Several researchers ([9, 10, 13]) have found that hybrid hierarchy is necessary when more fine-grained RBAC constraints, such as dynamic separation of duty (DSoD), temporal and cardinality constraints on roles in a hierarchy, need to be specified. Joshi *et al.* introduced three types of hierarchy by separating the permission inheritance semantics (in *I*-hierarchy type) and activation semantics (in *A*-hierarchy type). Roles related by *A*-hierarchy can be constrained by a DSoD constraint [6]. Joshi *et al.* also show that *A*-hierarchy is suitable for specifying permission-centric cardinality constraints on roles, while *I*-hierarchy or *IA*-hierarchy (which allows both permission and activation inheritance) is suitable for user-centric cardinality constraints. Furthermore, Du *et al.* and Shafiq *et al.* [2, 16] show that hybrid hierarchy is particularly useful when we want to integrate the multiple policies in multi-domain applications.

In our earlier work, we proposed a *Scoped Administration model for RBAC with Hybrid Hierarchy* (SARBAC07) model [17] that can be applied to RBAC model extended to support hybrid hierarchy. We refer to such an extend RBAC model as RBAC-HH in this paper. SARBAC07 is based on the idea introduced in the SARBAC model [1]. The contributions of SARBAC07 model include:

- (1) The SARBAC07 model can support administration operation in and RBAC with the hybrid hierarchy. It redefines concepts and operations of the SARBAC model.
- (2) The SARBAC07 model resolves an ambiguity in SARABC and show that the User-Role Assignment is determined by the *IA* -relation while the Role-Permission

Assignment is determined by the  $I$ -relation in a hybrid hierarchy.

In this paper, we propose a SARBAC-HH model that extends SARBAC07 with additional features to make it a more complete administrative model. In addition to the SARBAC07 model, this paper makes the following contributions:

- (1) We refine SARBAC07's Role Hierarchy Administration model (RHA07) and define more fine grained semantics for the administration operations. In particular, we show how each operation affects the overall role hierarchy and how the role hierarchy should be modified to keep the original permission acquisition relations in the RBAC-HH model.
- (2) We propose a family of RHA-HH models incrementally for the RBAC-HH model. The RHA-HH<sub>1</sub> model contains the basic elements for administering a hybrid hierarchy. The RHA-HH<sub>2</sub> model assigns administration permissions to each role. The RHA-HH<sub>3</sub> model introduces the special administrative roles. The RHA-HH<sub>4</sub> model adds the update features for the administrative roles. Note that the notion of the four models is similar to that of the SARBAC model but ours apply to the RBAC-HH model.
- (3) We compare our proposed SARBAC-HH model with the ARBAC07 model [18] that extends the ARBAC97 model and its successors for the RBAC-HH model. Crampton *et al.*'s SARBAC model has been shown to be more flexible than the ARBAC97 model [1]. We show that the SARBAC-HH model keeps this advantage over ARBAC07.

The rest of the paper is organized as follows. In Section 2, we review the relevant background and present the motivation for our work. We propose the family of RHA-HH model in Section 3. In Section 4, we introduce the complete SARBAC-HH model by adding the user-role assignment and permission role assignment administration models. We evaluate and compare our model with the ARBAC07 model in Section 5. We conclude our work in Section 6.

## 2. Background and Motivation

In this section, we briefly overview the hybrid hierarchy, introduce the SARBAC model, and present the motivation of our paper in section 2.3. We refer the readers to [4, 12] for details on the basic RBAC model.

### 2.1 Hybrid Hierarchy

Hybrid hierarchy was introduced in the context of the Generalized Temporal RBAC (GTRBAC) model to facilitate specifications of fine-grained policies [7]. In a hybrid hierarchy, a pair of roles may be related by one of the following three hierarchy types: *permission-inheritance-only* hierarchy ( $I$ -hierarchy, written as  $\geq_i$ ), *role-activation-only* hierarchy ( $A$ -hierarchy, written as  $\geq_a$ ) and the combined *permission-inheritance-activation* hierarchy ( $IA$ -hierarchy, written 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 who can activate  $s$  can also activate  $j$ . Figure 1 shows a sample hybrid hierarchy. Note that in the monotype 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 could 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 2(a):  $(x \langle f_1 \rangle y) \wedge (y \langle f_2 \rangle z) \rightarrow (x \langle f \rangle z)$ , where,  $(\langle f_1 \rangle \in \{\geq\}) \vee (\langle f_2 \rangle \in \{\geq\})$  and  $\langle f \rangle = \langle f_1 \rangle$ , if  $\langle f_2 \rangle \in \{\geq\}$ , otherwise  $\langle f \rangle = \langle f_2 \rangle$ .

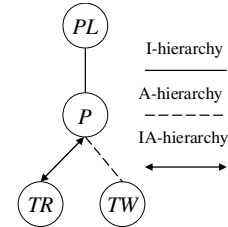


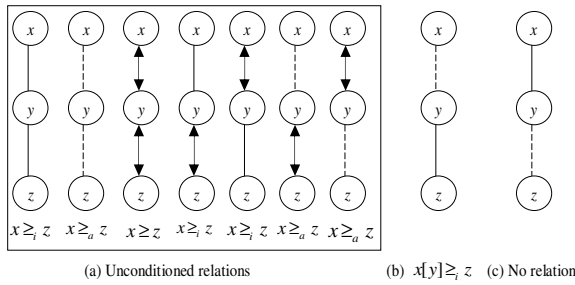
Figure 1. An example of hybrid hierarchy

A special case of derived relation is when an  $A$ -relation is followed by an  $I$ -relation, as shown in Figure 2(b), where the semantics is not straightforward. Here, by simply activating  $x$ , a user assigned to  $x$  can not acquire the permissions of  $z$ ; however, he can acquire the permissions of  $z$  by activating  $y$ . This means users assigned to  $x$  can still “acquire” the permissions of  $z$  even if there is neither  $I$  nor  $A$ -relation derived between  $x$  and  $z$ . In this case, we say that  $x$  has a “conditioned” relation with  $z$ , written as  $x[y] \geq_i z$ . We refer the readers to [8] for detailed treatment on the *conditioned derived* relation, where it 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 2(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 cannot activate  $y$ . Therefore, there is no derived relation between  $x$  and  $z$ . We define the derived relations as follows:

**DEFINITION 2.1 (Derived Relation):** Let  $x$  and  $y$  be roles such that  $(x \geq_a y)$ , 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].



**Figure 2.** Derived relations in a hybrid hierarchy

In standard hierarchy, if there is a path between role  $x$  and  $y$ , then  $x$  and  $y$  are hierarchical related since users assigned to  $x$  can acquire  $y$ 's permissions. In a hybrid hierarchy, however, this is not true if the path contains an  $I$ -relation followed by an  $A$ -relation. Therefore, we define the effective path in hybrid hierarchy as follows:

**DEFINITION 2.2 (Effective Path):** Given a hybrid hierarchy, we say a hierarchical path (denoted as  $p\langle x, y \rangle$ ) from a role  $x$  to a role  $y$  an “effective path” if and only if a user who can activate  $x$  can acquire the permissions available through  $y$ .

**THEOREM 2.1:** Path  $p\langle x, y \rangle$  is an effective path if there is no  $I$ -relation followed by  $A$ -relation in the path.

**PROPOSITION 2.1:** Path  $p\langle x, y \rangle$  is an effective path if  $x \geq_a y$ .

The proofs of Theorem 2.1 and proposition 2.1 follow directly from definition 2.1 and definition 2.2.

## 2.2 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. In this model, the notion of *administrative scope*, as defined below, is used to define which role can administer which roles.

**DEFINITION 2.3 (Administrative scope):** Given a role  $a$ , its administrative scope,  $S(a)$ , is defined as follows:

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

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 changes 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:  $AddRole(a, r, \Delta r, \nabla r)$ ,  $DeleteRole(a, r)$ ,  $AddEdge(a, c, p)$ , and  $DeleteEdge(a, c, p)$ , where  $\Delta r$  and  $\nabla r$  represent the sets of the immediate juniors and immediate seniors, respectively, of  $r$ . Table 2 summarizes the conditions that need to be satisfied for these operations to succeed. SARBAC defines a family of four RHA models, namely RHA<sub>1</sub>, RHA<sub>2</sub>, RHA<sub>3</sub>, and RHA<sub>4</sub>. The RHA<sub>1</sub> model is the simplest model. The operation is allowed to be performed if the conditions in Table 1 are satisfied. The RHA<sub>2</sub> model adds the administration permissions to each of the roles. Besides the conditions shown in Table 1, the administrative role must have the corresponding permissions to perform an

operation. The RHA<sub>3</sub> model creates a set of special administration roles and assigns each of them to the regular roles. In this way, each administration role can not only manage the regular roles assigned to it, but also all the roles within the administrative scopes of those regular roles. Finally, the RHA<sub>4</sub> model is the most complex model that discusses the administration of the administrative roles.

The operations and their success conditions for SARBAC-URA 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 section 4, we show some ambiguities associated with SARBAC-URA and SARBAC-PRA models.

**Table 1.** Hierarchy operations in SARBAC-RHA

Operation	Conditions
$AddRole(a, r, \Delta r, \nabla r)$	$\Delta r \subseteq S^+(a), \nabla r \subseteq S(a)$
$DeleteRole(a, r)$	$r \in S^+(a)$
$AddEdge(a, c, p)$	$c, p \in S(a)$
$DeleteEdge(a, c, p)$	$c, p \in S(a)$

**Table 2.** User-Role operations in SARBAC-URA

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

## 2.3 Motivation

The motivation for the use of *administrative scope* in the original SARBAC model is to ensure that no operation will cause undesirable “side-effect” to the other parts of the hierarchy. As per definition 2.2, if  $r \in S(a)$ , then every path upwards from  $r$  will eventually go through  $a$ . Therefore, all modifications  $a$  makes on  $r$  will only affect other roles also administered by  $a$ . In the presence of a hybrid hierarchy, the permission acquisition semantics become much more complex: even if there is a path from role  $x$  to role  $y$  the user assigned to  $x$  may not be able to access the permissions of  $y$ . Consider the hybrid hierarchy shown in Figure 1. Here, the role  $PL$  represents the leader of the programmer, the role  $P$  represents the programmer,  $TR$  represents the role that can only read the program, and  $TW$  represents the role that can only write the program. From Figure 1, we can see that the programmer can both read and write the program, and the leader can only read the program since  $PL$  and  $TW$  are connected by an  $I$ -hierarchy followed by an  $A$ -hierarchy. If we use the definition of administrative scope shown in definition 2.2, then  $TW \in S(PL)$  since all paths upwards from  $TW$  go through  $PL$ . However, using  $PL$  to administer  $TW$  may not be desirable since  $PL$  is not supposed to have the permissions assigned to  $TW$ . According to the policy,  $P$  is more suitable to administer  $TW$  as  $P$  can access all permissions of  $TW$ . Therefore, we see that while *administrative scope* is a flexible and effective notion to define administration relations in standard hierarchy, it cannot be used directly in the hybrid hierarchy. Motivated by this observation, we re-define the administrative scope and

other elements of SARBAC in SARBAC-HH to deal with the more complex hybrid hierarchy.

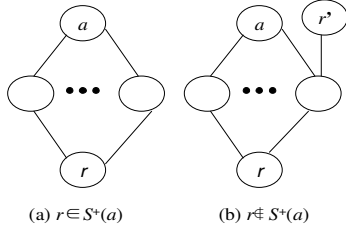


Figure 3. Administrative scope in SARBAC

### 3. The RHA-HH family

In this section, we propose a more comprehensive SARBAC-HH model. The Role Hierarchy Assignment model (RHA-HH) of SARBAC-HH focuses on the operations that modify one or more elements in the hybrid role hierarchy. We first re-define the administrative scope notion that is crucial in RHA-HH. We then define the administration operations supported by the RHA-HH. Finally, we propose four models of RHA-HH.

#### 3.1 Administrative Scope in RHA-HH

As discussed earlier, a role  $r$  can be administered under another role  $a$  iff all paths upwards from  $r$  go through  $a$ , as shown in Figure 3(a). On the contrary, suppose there is a path upwards from  $r$  that does not go through  $a$ , and instead, goes through role  $r'$ , as shown in Figure 3(b). Here  $a$  and  $r'$  are not hierarchical related, but both of them are hierarchical 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 is 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 the direct and indirect relations in the path between roles. 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  $S_{HH}^+(r) = S_{HH}(r) \setminus \{r\}$ . If  $r \in S_{HH}^+(a)$ , we call  $a$  as an administrator of  $r$ . According to the definition, if  $r \in S_{HH}(a)$ , then (1)  $r$  is junior to  $a$  in the hybrid hierarchy; (2) every effective path upwards from  $r$  also goes through  $a$ . Figure 4 illustrates the difference between the original administrative scope in SARBAC and the administrative scope in SARBAC-HH. Note that the structure of the three hierarchies is exactly the same and the only difference is the type of the hierarchy. 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 is no derived relation between  $r$  and  $r'$  even if there is 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 show that our definition of administrative scope provides better flexibility and maintains the decentralization/autonomy properties.

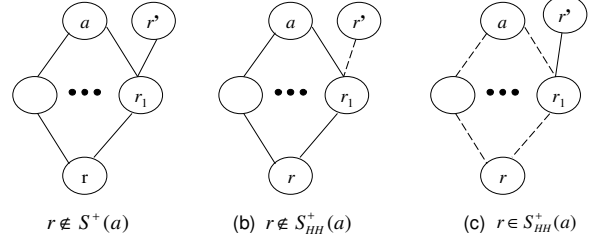


Figure 4. Administrative Scope in SARBAC and SARBAC-HH

**Flexibility:** Administrative scope in both the models is determined by the role hierarchy itself, and changes dynamically as the hierarchy changes. 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 theorem. We retain the notion of the *line manager* from the SARBAC model. Intuitively, the *line manager* is the minimum administrator for all administrators of a role. Line manager can be used to ensure maximum decentralization and accountability. That is, we can insist that all changes affecting a role are made by the line manager.

**THEOREM 3.1 (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  $z \leq_d y$  implies  $z = y$ ; hence,  $x \not\leq_d y$  and  $y \not\leq_d x$ . Then,  $r \in S_{HH}^+(x)$  and hence  $x \in \uparrow r$ . Similarly,  $r \in S_{HH}^+(y)$  and hence  $\uparrow r \setminus \uparrow y \subseteq \downarrow y$ .  $x \notin \uparrow y$  gives  $x \in \downarrow y$ . Thus,  $x < y$ , which is a contradiction. ■

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

The original administrative scope in SARBAC is shown to have “nested” property [1]. That is, an administrative scope is either completely inside another administrative scope or disjoint from other administrative scopes. This property is not maintained in RHA-HH. Consider the hybrid hierarchy shown in Figure 5. According to the definition 3.1, we have:

$$S_{HH}(r_0) = \{r_1, r_2, r_3, r_4, r_5, r_7, r_8, r_9, r_{10}, r_{12}, r_{16}, r_{17}, r_{18}\},$$

$$S_{HH}(r_1) = \{r_4, r_5, r_6, r_{12}, r_{13}\}, S_{HH}(r_2) = \{r_7, r_8\},$$

$$S_{HH}(r_3) = \{r_9, r_{10}, r_{18}\}, S_{HH}(r_4) = \{r_{11}\}, S_{HH}(r_{10}) = \{r_{18}\}$$

$$S_{HH}(r_5) = S_{HH}(r_6) = S_{HH}(r_7) = S_{HH}(r_8) = S_{HH}(r_9) = S_{HH}(r_{11}) = S_{HH}(r_{12}) = S_{HH}(r_{13}) = S_{HH}(r_{14}) = S_{HH}(r_{15}) = S_{HH}(r_{16}) = S_{HH}(r_{17}) = S_{HH}(r_{18}) = \emptyset$$

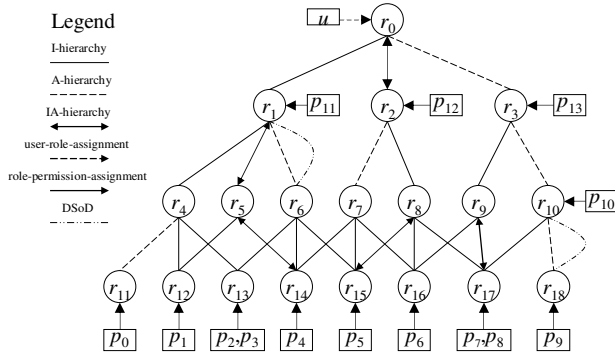


Figure 5: An example of the hybrid hierarchy

The administrative scopes  $S_{HH}(r_0)$  and  $S_{HH}(r_1)$  are neither nested nor disjoint. In fact, they are overlapped. Next, we show several useful properties of the *administrative scope* in the hybrid hierarchy.

**THEOREM 3.2:**  $\forall x, y, x_c, y_c \in R$  such that  $x \succeq_d x_c, y \succeq_d y_c$

Then the following holds:

$$(x \succ_d y) \wedge (y \succ_d x) \rightarrow (x_c \notin S_{HH}(y)) \wedge (y_c \notin S_{HH}(x))$$

**PROOF:** We first prove that if both  $x \succ_d y$  and  $y \succ_d x$  hold, then  $x_c \notin S_{HH}(y)$  also holds. To prove  $x_c \notin S_{HH}(y)$ , we need to show that either  $y \succ_d x_c$  or there is an effective path going upwards from  $x_c$  that does not go through  $y$ . Since  $x \succeq_d x_c$ , there is an effective path going upwards from  $x_c$  through  $x$ . Next, we need to show that  $y$  is not on this path. We prove it by contradiction. If  $y$  is between  $x_c$  and  $x$ , that is,  $x \succeq_d y \succeq_d x_c$ , it contradicts with  $x \succ_d y$ ; If  $y$  is senior to  $x$ , that is,  $y \succeq_d x \succeq_d x_c$ , it contradicts with  $y \succ_d x$ . Therefore, there is an effective path upwards from  $x_c$  but does not go through  $y$ , and we get  $x_c \notin S_{HH}(y)$ . We can prove that  $y_c \notin S_{HH}(x)$  in the similar way. ■

Theorem 3.2 shows that if two roles  $x$  and  $y$  are not hierarchically related to each other then all the children (either immediate or derived) of  $x$  cannot be administered by  $y$  and all the children of  $y$  (either immediate or derived) cannot be administrated by  $x$ . This property is straight forward in the real life. If two leaders' jobs are not related in an organization, they should not administer each other's staff members' privileges. In Figure 5,  $r_0$  and  $r_6$  are not hierarchically related, so all of  $r_6$ 's children -  $r_{13}$ ,  $r_{14}$ , and  $r_{15}$  - are not in the *administrative scope* of  $r_0$ .

**THEOREM 3.3:**  $\forall r, a \in R$ , let  $\nabla_a r, \nabla_i r$  be the set of all immediate A-senior and immediate I-senior roles, respectively, of  $r$ , then the following holds:

$$(\forall s \in \nabla_a r \cup \nabla_i r, s \in S_{HH}(a)) \rightarrow r \in S_{HH}(a)$$

**PROOF:** from  $(\forall s \in \nabla_a r \cup \nabla_i r, s \in S_{HH}(a))$ , we have every effective path upwards from the role set  $\nabla_a r \cup \nabla_i r$  go through  $a$ . Since  $\nabla_a r \cup \nabla_i r$  represents all immediate senior roles of  $r$ , every effective paths upwards from  $r$  also goes through  $a$ . therefore, we get  $r \in S_{HH}(a)$ . ■

Theorem 3.3 shows that if every immediate senior roles of a role  $r$  is within the administrative scope of the role  $a$ , then  $r$  is also within the administrative scope of  $a$ .

### 3.2 Operations in RHA-HH

As discussed before, the operations in RHA-HH are used to modify the roles and hybrid hierarchies. All operations supported in RHA-HH and their success conditions are shown in Table 3. Here,  $\Delta_a r$  is the set of immediate A-juniors of role  $r$ ,  $\nabla_a r$  is the set of immediate A-seniors of role  $r$ ,  $\Delta_i r$  is the set of immediate I-juniors of role  $r$ , and  $\nabla_i r$  is the set of immediate I-seniors of role  $r$ , as shown in Figure 6. Compared to the original SARBAC-RHA model shown in Table 1, the RHA-HH adds two new operations: *PartitionRole* and *ChangeEdge*.

Ideally, after each operation, we should keep the original semantics as much as possible. For example, when we want to delete a role  $r$ , which has an immediate senior  $s$  and an immediate junior  $j$ , we need to maintain the original relation between  $s$  and  $j$  after the operation. Moreover, to make sure same users can acquire same permissions after deleting the role, we may need to reassign permissions of  $r$  to other roles and reassign users of  $r$  to other roles. This is a challenging problem and is beyond the scope of this paper. Interested readers are referred to [8], where Joshi *et al.* analyze these issues in more detail. In the rest of this section, we will focus on maintaining the *administrative scope* during those operations. Specifically, the following two criteria need to be satisfied:

$C_1$ : After each operation, the new role(s) (if any) should be within the administrative scope of  $a$ .

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

Next, we discuss the success conditions and discuss whether each operation satisfies criteria  $C_1$  and  $C_2$ .

Table 3: Operations and their success conditions in RHA-HH

Operation	Success Conditions
$AddRole(a, r, \Delta_a r, \nabla_a r, \Delta_i r, \nabla_i r)$	$\Delta_a r \subseteq S_{HH}^+(a) \nabla_a r \subseteq S_{HH}(a)$ $\Delta_i r \subseteq S_{HH}^+(a) \nabla_i r \subseteq S_{HH}(a)$
$DeleteRole(a, r)$	$\Delta_a r \subseteq S_{HH}^+(a) \nabla_a r \subseteq S_{HH}(a)$ $\Delta_i r \subseteq S_{HH}^+(a) \nabla_i r \subseteq S_{HH}(a)$
$PartitionRole(a, r)$	$\Delta_a r \subseteq S_{HH}^+(a) \nabla_a r \subseteq S_{HH}(a)$ $\Delta_i r \subseteq S_{HH}^+(a) \nabla_i r \subseteq S_{HH}(a)$
$AddEdge(a, j, s, type)$	$j, s \in S_{HH}(a)$
$DeleteEdge(a, j, s)$	$j, s \in S_{HH}(a)$
$ChangeEdge(a, j, s, type)$	$j, s \in S_{HH}(a)$

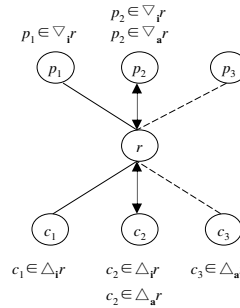


Figure 6: Parameters in AddRole

#### 3.2.1 AddRole operation

In a hybrid hierarchy, we need to be able to identify the A-senior roles, A-junior roles, I-senior roles and I-junior roles

of any role. Note that an *IA*-senior/junior role can be deemed as both an *I*-senior/junior and an *A*-senior/junior role. All these roles should be within the *administrative scope* of  $a$ , otherwise, the operation performed by  $a$  (e.g. *AddRole*) may cause unexpected side-effect (e.g. have a new senior/junior role) on those roles. Moreover, we note that the senior roles of  $r$  could be  $a$  itself but the junior roles cannot. Therefore, we require that the senior roles should be within  $S_{HH}(a)$  while the junior roles should be within  $S_{HH}^+(a)$ .

It is straightforward that the *AddRole* operation satisfies  $C_1$  according to Theorem 3.3, since all immediate senior roles of  $r$  are within the administrative scope of  $a$ . For  $C_2$ , we only need to show that all the immediate junior roles' administrators are not changed, since the administrator of a role  $r$  only depends on its senior roles.  $\forall j \in \Delta_{ar} \cup \Delta_{ir}$ ,  $a$  is the administrator of  $j$  according to Table 3, which means every effective path upwards from  $j$  goes through  $a$ . After adding the role, there is one more effective path upwards from  $j$  (i.e. the path goes through the new role  $r$ ). It is obvious that this effective path also goes through  $a$ . Therefore,  $a$  is still the administrator of  $j$  after adding the new role. If  $j$  has other administrators, say  $a'$ , we have  $a \succeq_a a'$  or  $a' \succeq_a a$  according to theorem 3.2. Similarly, we can easily prove that  $a'$  is still the administrator of  $j$  after adding the new role.

### 3.2.2 DeleteRole operation

When deleting a role  $r$ , we also need to make sure all the immediate senior and junior roles of  $r$  are within the administrative scope of  $a$ . Otherwise, the operation carried out by role  $a$  (e.g. *DeleteRole*) will cause unexpected side-effect (e.g. a senior/junior role is deleted) on those roles.  $C_1$  does not apply to *DeleteRole* operation since it does not introduce new roles. Note that *DeleteRole* does not always satisfy  $C_2$ . Consider an immediate junior role  $j$  of  $r$  (either *I*-junior or *A*-junior). Since  $j \in S_{HH}(a)$ , there is at least one effective path from  $a$  to  $j$ . If there is only one such effective path, then  $r$  must be on this path too. After deleting the role  $r$ ,  $a$  and  $j$  are not connected. Therefore,  $j \notin S_{HH}(a)$  after deleting the role, and hence  $C_2$  is violated.

### 3.2.3 PartitionRole operation

The semantic of *PartitionRole* is complex and was first introduced in [8]. Specifically, we can partition a given role vertically, horizontally, or by using a combination of both. How to keep the original permission acquisition relations after partitioning a role is discussed in [8] in more detail. From the administration perspective, we need to make sure all the immediate senior and junior roles of  $r$  are within the administrative scope of  $a$ , as is with the *AddRole* and *DeleteRole* operations.

Similar to *AddRole* operation, *PartitionRole* operation also satisfies  $C_1$ . This is because *PartitionRole* can be deemed as first deleting  $r$  then adding the new roles, and neither *DeleteRole* nor *AddRole* violates  $C_1$ . For  $C_2$ , as long as the junior roles of  $r$  are connected to at least one new role partitioned from  $r$ , the administrators of those junior roles are not changed. The proof is similar to that for the *AddRole* operation.

### 3.2.4 AddEdge operation

*AddEdge* operation is used to add a hierarchical edge between previously unrelated roles. Let  $j$  refer to the junior role after adding a new edge and  $p$  refer to the senior role after adding the edge, and *type* is one of the following: *A*-hierarchy, *I*-hierarchy, or *IA*-hierarchy. Obviously this operation will affect  $j$  and  $s$ . Therefore, we need to make sure both  $j$  and  $s$  are within the administrative scope of  $a$ .

$C_1$  does not apply to *AddEdge* since no new roles are added.  $C_2$  is not always satisfied after adding the edge. One counter-example is when  $s$  and  $j$  are isolated roles (not hierarchical related to any other roles) before adding the edge. In this case,  $j$  has no administrator besides itself before adding the edge but have one administrator ( $s$ ) besides itself after adding the edge.

### 3.2.5 DeleteEdge operation

*DeleteEdge* operation is used to delete an existing hierarchical relation between the senior role  $s$  and junior role  $r$ . Obviously, both  $s$  and  $r$  need to be within the *administrative scope* of  $a$  to prevent un-expected side effects.

$C_1$  does not apply to *DeleteEdge* since no new roles are added.  $C_2$  is not always satisfied after deleting the edge. One counter-example is when  $s$  and  $j$  are only hierarchical related to each other (do not connect to any other roles). In this case,  $j$  has one administrator besides itself before deleting the role ( $s$ ) but have no administrator besides itself after deleting the role.

### 3.2.6 ChangeEdge operation

*ChangeEdge* operation is used to change the hierarchical relation between two previously related roles. Again, both the senior role  $s$  and junior role  $r$  need to be within the administrative scope of  $a$ .

$C_1$  does not apply to *ChangeEdge* as no new roles are added.  $C_2$  is satisfied after the *ChangeEdge* operation. This conclusion is not straight-forward. *ChangeEdge* can be deemed as first deleting the edge and then adding a new edge and both of these two operations do not always satisfy  $C_2$ . However,  $C_2$  is satisfied after combining these two operations. Consider the hybrid hierarchy shown in Figure 7(a) where  $r \in S_{HH}(a)$ . If we change the edge  $(r, r_1)$  to the *I*-type, as Figure 7(b) shows,  $r \notin S_{HH}(a)$  now. However, in Figure 7(a),  $r_1$  is not administered by  $a$ , so the *ChangeEdge* operation fails. In other words, if *ChangeEdge* operation succeeds, it is guaranteed that it will not affect the administrators of all the original roles. Therefore,  $C_2$  is satisfied. Next, we introduce the family of RHA-HH models based on these operations.

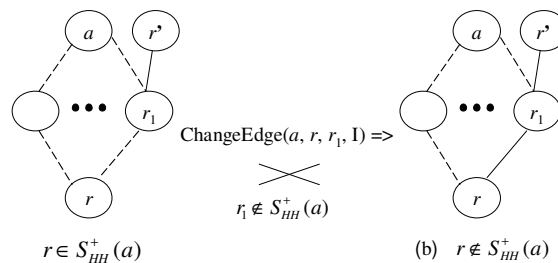


Figure 7. The ChangeEdge operation will not succeed

### 3.3 RHA-HH<sub>1</sub>

The RHA-HH<sub>1</sub> model is the simplest model in the RHA-HH family. In RHA-HH<sub>1</sub>, as long as the condition in Table 3 is satisfied,  $a$  can perform the corresponding operation to update the role hierarchy. For example, in Figure 5,  $r_4$  can change the  $A$ -edge  $\langle r_4, r_{11} \rangle$  to an  $I$ -edge, since  $r_4 \in S_{HH}(r_4)$  and  $r_{11} \in S_{HH}(r_4)$ . The basic RHA-HH<sub>1</sub> cannot support more fine-grained administration requirements. For example,  $P_2$  is not always satisfied as discussed before. To solve this problem, we can further require that only those operations that satisfy  $P_2$  be allowed to be performed. As a result, *DeleteRole*, *AddEdge*, *DeleteEdge* operations cannot always be performed as they do not satisfy  $P_2$ . For these kinds of operations, only the specific operations that satisfy  $P_2$  can be performed.

### 3.4 RHA-HH<sub>2</sub>

The RHA-HH<sub>2</sub> model requires that, in addition to the success condition shown in Table 3,  $a$  also have corresponding administration permissions assigned to it in order to perform hierarchy transformation operations. This is useful when more than one administrative role can perform the same operation and we want to divide the responsibilities of each administrative role. For example, in Figure 5, both  $r_1$  and  $r_4$  are the administrators of  $r_{11}$ . To support fine-grained administration, we can assign the permissions related to *AddEdge*, *DeleteEdge*, *ChangeEdge* to  $r_4$ , and assign the permissions related to *AddRole*, *DeleteRole*, *PartitionRole* to  $r_1$ . As a result,  $r_4$  can perform the operations related to hierarchical edges and  $r_1$  can perform all the operations (note that  $r_1 \succeq_d r_4$  so  $r_1$  also have permissions to perform edge operations). The administration of such permission-role assignment is defined in the SARBAC-HH-PRA model, as shown in Section 4.

### 3.5 RHA-HH<sub>3</sub>

In RHA-HH<sub>3</sub>, we define a relation called *admin-authority*  $\subseteq A \times R$ , where  $A$  is the set of administrative roles and  $R$  is the set of all regular roles. If  $(a, r) \in \text{admin-authority}$ , then  $a$  is called the administrative role and can administer the regular roles within the *administrative scope* of  $a$ . As a result, we modify the success condition in Table 3 as follows:

“there exists at least one regular role  $r'$  such that  $(a, r') \in \text{admin-authority}$  and all those corresponding roles are within the *administrative scope* of  $r'$ .”

For example, if we associate an administrative role  $a$  to the role  $r_4$  in Figure 5, that is,  $(a, r_4) \in \text{admin-authority}$ , then  $a$  can delete the edge  $\langle r_4, r_{11} \rangle$  since both  $r_4$  and  $r_{11}$  are within the *administrative scope* of  $r_4$  and  $a$  is associated with  $r_4$  in *admin-authority*.

Note that the *admin-authority* relation is also used in the original SARBAC model. In summary, besides the ability to support hybrid hierarchy, our RHA-HH<sub>3</sub> model has two advantages over the original SARBAC-RHA model:

- (1) SARBAC-RHA modifies the definition of *administrative scope* to define which administrative role can perform operations on which regular roles according to *admin-authority* [x] while RHA-HH<sub>3</sub> keeps the definition of *administrative scope* and only modifies the success conditions shown in Table 3. In our model, we use

*admin-authority* to relate an administrative role  $a$  to a regular role  $r$  and let  $a$  administer all roles in  $S_{HH}(r)$ . Therefore, we believe it is more straight-forward to keep the *administrative scope* and only modify the success conditions of each operation.

- (2) SARBAC-RHA treats the *admin-authority* as an extended role hierarchy. That is,  $a$  is made senior to  $r$  in the extended hierarchy if  $(a, r) \in \text{admin-authority}$ . However, we believe that using extended hierarchy does not make sense since  $a$  does not inherit any permission from  $r$ . In our model, we distinguish the administrative role set  $A$  and the regular role set  $R$  clearly, and relate them only through the *admin-authority* relation.

### 3.6 RHA-HH<sub>4</sub>

In RHA-HH<sub>4</sub>, we introduce the administration model to administer the *admin-authority* itself. First, we note that the administrative role set  $A$  itself can form a hybrid hierarchy. That is,  $a_1 \succeq_a a_2$  means that the users assigned to  $a_1$  can activate  $a_2$  to perform all the operations that  $a_2$  can perform;  $a_1 \succeq_i a_2$  means that the users assigned to  $a_1$  can perform all the operations that  $a_2$  can perform; and  $a_1 \succeq a_2$  indicates both  $a_1 \succeq_a a_2$  and  $a_1 \succeq_i a_2$ . By using the hybrid hierarchy over the administrative roles, the advantages of hybrid hierarchy such as supporting the user-centric and permission-centric SoDs can also be applied to the administrative roles.

An important issue is identifying who can define and modify the *admin-authority* relation and who can modify the hybrid hierarchy over  $A$ . We note that the administrative scope of each administrative role can be defined in the same way as that of the regular roles. Based on this, we define a new operation: *AddAdminAuthority*( $a', a, r$ ), where  $a'$  is the administrator to perform this operation and  $a$  and  $r$  are the administrative roles and regular roles that need to be hierarchically related. Note that  $a'$  itself must belong to administrative roles but not regular roles. The success condition of this operation is  $a' \in S_{HH}(a')$  and  $(a', r) \in \text{admin-authority}$ . In other words,  $a'$  should be able to administer both  $a$  and  $r$  before it can add an *admin-authority* relation between them. For the modification of the hybrid hierarchy over  $A$ , we use the same operations and success conditions as in Table 3. The only difference is that all the roles are administrative roles.

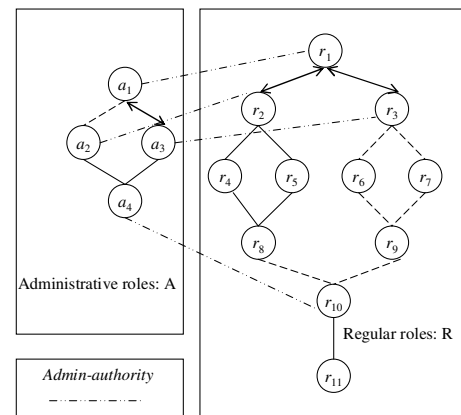


Figure 8: An example of admin-authority relations

Figure 8 shows an example of administrative roles  $A$ , regular roles  $R$ , and the *admin-authority* relations. As shown in Figure 8,  $a_1$  can operate on the role set  $\{r_2, r_4, r_5, r_8\}$  since it is contained in the administrative scope of  $r_1$  and  $(a_1, r_1) \in \text{admin-authority}$ . Similarly,  $a_1$  can not operate on  $r_{10}$  but  $a_4$  can. However, the administrator assigned to  $a_1$  can inherit the permissions of  $a_4$  through the  $I$ -hierarchy. Therefore, they can also operate on  $r_{10}$ . Besides,  $a_1$  can also remove  $(a_3, r_3)$  from *admin-authority* since  $(a_1 \geq a_3)$  and  $a_1$  can administer  $r_3$ . Finally,  $a_1$  can also change the edge  $\langle a_1, a_3 \rangle$  from the  $IA$ -edge to the  $I$ -edge, since both  $a_1$  and  $a_3$  are within the *administrative scope* of  $a_1$  in  $A$ .

#### 4. SARBAC-HH

In this section, we introduce the complete SARBAC-HH model by adding the user-role assignment model (URA-HH) and the permission-role assignment model (PRA-HH) to the RHA-HH model. The structure of the entire SARBAC-HH model is shown in Figure 9.

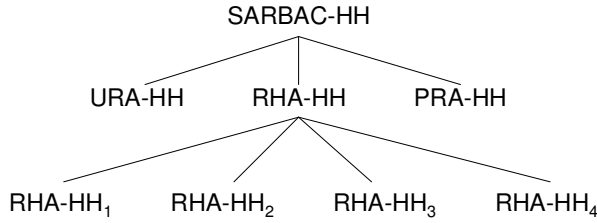


Figure 9: the entire structure of the SARBAC-HH model

The key operations in SARBAC-URA are shown in Table 2. The permission-role assignment operations in SARBAC-PRA are similar. We first show that there is an ambiguity in the semantics of user-role assignment and permission-role assignment in the original SARBAC. We then show that our model can remove this ambiguity by redefining those operations in presence of hybrid hierarchy. To illustrate these operations, we first review the notion of constraint in SARBAC. Let  $R' = \{r_1, \dots, r_k\}$  be a subset of  $R$  and let  $\wedge R'$  denote  $r_1 \wedge \dots \wedge r_k$ .

**DEFINITION 4.1 (SARBAC constraint):** *The 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  $C \subseteq \uparrow R(p)$ ; here for any  $Y \subseteq X$ ,  $\uparrow Y = \{x \in X : \exists y \in Y \text{ such that } x \geq y\}$ , and  $\downarrow Y = \{x \in X : \exists y \in Y \text{ such that } x \leq y\}$ .*

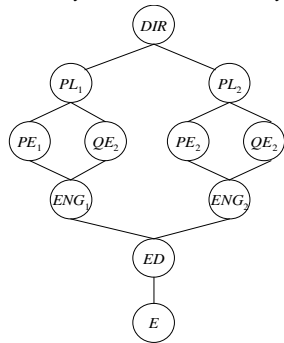


Figure 10: A monotype hierarchy

We first analyze under what situation a user will satisfy a

constraint. An example of monotype hierarchy is shown in Figure 10, which was used in the original SARBAC paper [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  $QE_1$ , thus the  $PE_1 \wedge QE_1$  constraint is satisfied. Obviously, the author of SARBAC implicitly assumes the hierarchy relation in any monotype hierarchy as an “Is-a” relation [10]. That is,  $x \geq y$  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 monotype hierarchy have long been argued as ambiguous [9, 10, 13]. The hierarchical relation in a monotype 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 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 4.1, we re-define it as follows:

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

Note that the definition looks same as before, but here the symbol  $\leq$  clearly means the  $IA$ -relation in hybrid hierarchy. Next we analyze in which cases a permission will satisfy a constraint. In Figure 10, according to Definition 4.1 the constraint  $PE_1 \wedge QE_1$  is satisfied by any permission assigned to both  $PE_1$  and  $QE_1$ , and by those 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  $QE_1$ , thus satisfying the  $PE_1 \wedge QE_1$  constraint. 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 monotype 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 monotype hierarchy, as claimed by many researchers [9, 10, 13]. Again, the use of hybrid hierarchy can provide clearer semantics by using explicitly “ $I$ -relation”. Note that whether a permission satisfies a constraint depends on the definition of  $\uparrow Y$  in Definition 4.1; hence, we re-define  $\uparrow Y$  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 define the SARBAC-HH constraint as follows:

**DEFINITION 3.4 (SARBAC-HH constraint):** *An SARBAC-HH constraint has the form  $\wedge C$  for some  $C \subseteq R$ . Constraint  $\wedge C$  is satisfied by a user  $u$  if  $C \subseteq \downarrow R(u)$ .  $\wedge C$  is satisfied by a permission  $p$  if  $C \subseteq \uparrow R(p)$ . Here, the symbol  $\uparrow$  and  $\downarrow$  are as defined by (1) and (2).*

The definition implies that the User-Role Assignment is determined by the  $IA$ -relation in the hybrid hierarchy, while

the Permission-Role Assignment is determined by the  $I$ -relation in the hybrid hierarchy. The user-role assignment operations are the same as in SARBAC, as shown in Table 3 and permission-role assignment operations are similar.

## 5. Evaluation

In this section, we will evaluate our SARBAC-HH model carefully using the same criteria in the original SARBAC paper in section 5.1 to section 5.4. We then compare our model with ARBAC07 [18] – that extends ARBAC97 model – we have proposed earlier and can deal with the hybrid hierarchy either in section 5.5.

### 5.1 Availability

For an administration model, *availability* means any administration model should allow as many necessary operations as possible [1]. For example, if a role  $a$  is supposed to administer a role  $r$ , then  $a$  should be able to delete role  $r$ . On the other hand, almost all administration models add some constraints on certain operations to prevent “illegal operations”. For example,  $DeleteEdge(a, j, s)$  requires that  $j, s \in S_{HH}(a)$ . The more constraints we add on, the less *availability* we have. Crampton *et al.* [1] have claimed that SARBAC addresses better *availability* than the ARBAC97 model does, since a lot of necessary operations that are not allowed in ARBAC97 are allowed in SARBAC [1]. In Section 3, we showed that the major difference between original SARBAC and our SARBAC-HH model is the definition of the administrative scope concept as well as the operations. So we believe that our model supports *availability* at least upto the same extent as does the original SARBAC model.

### 5.2 Robustness

Similar to the SARBAC model, all the elements in our model, including  $R, RH, UA, PA, ua$ -constraints,  $pa$ -constraints, and *admin-authority*, are dynamic in nature. If the role hierarchy changes, the administrative scope will change dynamically according to the new hierarchy. And the semantics of those operations will also change dynamically. This shows that our model is *robust* to any changes in the hierarchy.

### 5.3 Simplicity

Obviously, the use of hybrid hierarchy will increase the complexity of the administration model significantly. However, we believe this complexity is because of the complex nature of the hybrid hierarchy, and not because of our model. Specifically, we only re-define the concept of administrative scope and some operations to include the hybrid hierarchy into our model.

### 5.4 Practicality and Versatility

We will use two examples to show our SARBAC-HH model and demonstrate that it is better in terms of both *practicality* and *versatility*. Also note that the SARBAC is inadequate for the hierarchies in the examples.

**Example 4.1:** Consider 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 that  $P$  has carried out. Figure 1 depicts the hierarchy that can be generated for

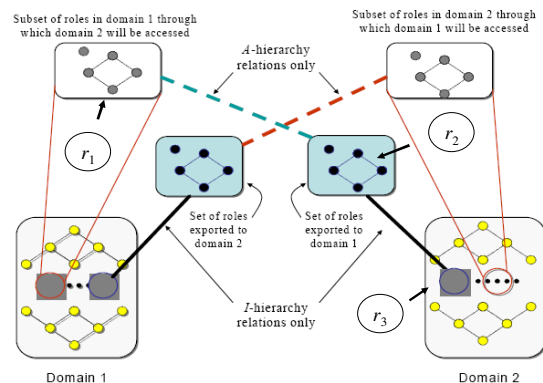
achieving the goal. Role  $TaskR$  ( $TR$ ) contains the read-only permissions whereas role  $TaskW$  ( $TW$ ) contains all the write/modify permissions related to the programming task. The role  $PL$  is  $I$ -senior to the role  $P$ . Note that users assigned to role  $PL$  can acquire permissions of  $TR$  but not of  $TW$ .

In this example, monotype hierarchy is inadequate. Since  $PL$  only has the read permissions over the program code created and not the permission to edit the code, we need two separate roles such as  $TR$  and  $TW$ . If we use monotype 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 should use hybrid hierarchy as shown in Figure 1 to ensure the accurate semantics.

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

**Example 4.2:** Assume domain 1 and domain 2 both require services (a set of permissions) from each other. In an 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.

In this example, monotype hierarchy will not work. We should use  $I$ -relations and  $A$ -relation as in Figure 11 to prevent the transitivity of the activation semantics which is usually the underlying problem in inter-domain access [2].



**Figure 11:** Inter-domain role mapping scenario

In figure 11, although  $r_1$  can have the permission assigned to  $r_3$  by activating  $r_2$ ,  $r_1$  cannot 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

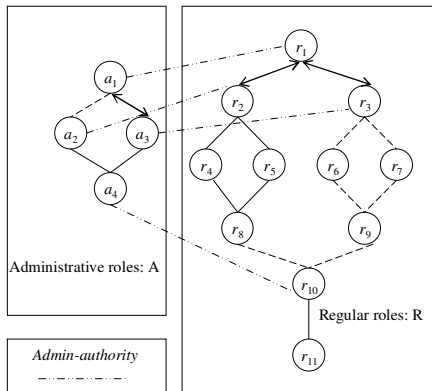
domain 1 can only administer the roles “specially exported” from domain 2 but cannot administer the “local” roles in domain 2.

We cannot enumerate all possible situations, as can be seen from these two different examples we believe that our model is both practical and versatile.

### 5.5 Comparison between SARBAC-HH and ARBAC07

In this section, we compare our proposed SARBAC-HH model with the ARBAC07 model. The ARBAC07 model extends the well-known ARBAC97 model for the RBAC-HH model. The SARBAC model is claimed to be more flexible and simpler than the ARBAC97 model. We show that our SARBAC-HH model also have advantages over the ARBAC07 model.

First, we briefly review the ARBAC07 model. We only present the parts that are directly relevant for the comparison here. Like ARBAC97, ARBAC07 also have three sub-models: User-Role Assignment (URA07), Permission-Role Assignment (PRA07), and Role-Role Administration (RRA07). In URA07, a user  $u$  is said to satisfy a precondition  $r$  (the precondition is in the form of a regular role) if and only if  $u$  is a member of  $r$  or the  $IA$ -senior of  $r$ ; and the user-role assignment is specified as  $can\_assign(x, y, z)$ , which means a member of the administration role  $x$  (or its senior) can assign the users who satisfy the precondition  $y$  to the role range  $z$ . In PRA07, a permission  $p$  is said to satisfy a precondition  $r$  if and only if  $p$  is assigned to  $r$  or  $p$  is assigned to  $r$  or the  $I$ -junior of  $r$ ; and the permission-role assignment is specified as  $can\_assignp(x, y, z)$ , which means a member of the administration role  $x$  (or its senior) can assign the permissions who satisfy the precondition  $y$  to the role range  $z$ . In RRA07, the role-role administration is specified as  $can\_modify(x, y)$ , which means a member of the administration role  $x$  can modify the encapsulated role range  $y$ . A role range is said to be encapsulated if it has exactly one senior most role and one junior most role. Furthermore, RRA07 requires that the encapsulated range is kept after the role hierarchy operation.



**Figure 12** An example of admin-authority relations

We use the example shown in Figure 8 (re-arrange it here as Figure 12 for the ease to read) to compare ARBAC07 and SARBAC-HH. The admin-authority relations used in SARBAC-HH is shown in figure 12. To be consistent, we add the following  $can\_modify$  relation to the ARBAC07 policy:  $can\_modify(a_1, (r_1, r_{11}))$ ,  $can\_modify(a_2, (r_2, r_8))$ ,

$can\_modify(a_3, (r_3, r_9))$ , and  $can\_modify(a_4, (r_{10}, r_{11}))$ . It is easy to verify that the role ranges in all these relations are *encapsulated*. For simplicity, we use  $u_n$  to refer to the user assigned to  $r_n$  ( $n=1, \dots, 11$ ). Table 4 shows whether a certain operation is supported or not in ARBAC07 and SARBAC-HH. In Table 4 we specify the operation in English rather than in formal language since the two models use different specification languages. We say a model supports the operation if all the success conditions for the operation in the model are satisfied. For example, the operation “ $a_1$  add role  $x$  between  $r_1$  and  $r_5$ ” is not supported by ARBAC07 since the *encapsulated role range*  $(r_2, r_8)$  is violated after such an operation. However, the same operation is supported by SARBAC-HH since  $r_1 \in S_{HH}(r_1)$ ,  $r_5 \in S_{HH}(r_1)$  and  $(a_1, r_1) \in admin\_authority$ . The other operations can be explained similarly.

From table 4, we can see that SARBAC-HH supports much wider set of operations than ARBAC07. This conclusion is similar to the comparison between SARBAC and ARBAC97. Therefore, we show that our model keeps the main advantage of SARBAC over ARBAC07 and provides much more fine-grained administration semantics by using the hybrid hierarchy.

## 6 Related Work

Several researchers have studied how to use role itself to manage RBAC policies and several models have been proposed. Sandhu *et al.* [14] propose an ARBAC97 model consisting of URA97 (User-Role Assignment), PRA97 (Role-Permission Assignment) and RRA97 (Role Hierarchy Administration). URA97 define  $can\_assign$  and  $can\_revoke$  relations to manage the  $AssignUser$  and  $AssignPermission$  operations, and PRA97 is a counterpart of URA97. The fundamental idea in RRA97 is an encapsulated range, which is a “closed” sub-hierarchy that every path upwards goes through the same role. The  $can\_modify$  relation define which administration role can manage which encapsulated range. They further extend ARBAC97 to ARBAC99 and later to ARBAC02, but the key idea is still the same. Crampton *et al.* have proposed a SARBAC model [1] motivated by the shortcomings of the ARBAC97 model. The key idea is the administrative scope, which is similar to the encapsulated range. Both of these two models use monotype hierarchy.

Table 4: Operations in ARBAC07 and SARBAC-HH (using the policies shown in figure 12)

ARBAC07	SARBAC-HH
$a_1$ add role $x$ and between $r_1$ and $r_5$ using $I$ -hierarchy	
Not supported, because: the encapsulated range $(r_2, r_8)$ is violated	Supported, because: $r_1 \in S_{HH}(r_1)$ , $r_5 \in S_{HH}(r_1)$ , and $(a_1, r_1) \in admin\_authority$
$a_2$ delete role $r_8$	
Not supported, because: $can\_modify(a_2, (r_2, r_8))$ and $r_8 \notin (r_2, r_8)$	Supported, because: $r_8 \in S_{HH}(r_2)$ and $(a_2, r_2) \in admin\_authority$
$a_1$ partition role $r_2$ into $r_{21}$ and $r_{22}$ vertically using $I$ -hierarchy	
Not supported, because: $can\_modify(a_1, (r_1, r_{11}))$ , $r_2 \in (r_1, r_{11})$ , and no	Supported, because: $r_2 \in S_{HH}(r_1)$ and $(a_1, r_1) \in admin\_authority$

encapsulated role range is violated	
$a_1$ add an A-edge between $r_1$ and $r_5$	
Not supported, because: the encapsulated range ( $r_2, r_8$ ) is violated	Supported, because: $r_1 \in S_{HH}(r_1), r_5 \in S_{HH}(r_1) (a_1, r_1) \in admin-authority$
$a_1$ delete the edge ( $r_8, r_{10}$ )	
Not supported, because: the encapsulated range ( $r_2, r_8$ ) is violated	Supported, because: $r_8 \in S_{HH}(r_1), r_{10} \in S_{HH}(r_1),$ and $(a_1, r_1) \in admin-authority$
$a_2$ change the edge ( $r_5, r_8$ ) to the A-edge	
Supported, because: $can\_modify(a_2, (r_2, r_8))$	Supported, because: $r_5 \in S_{HH}(r_2), r_8 \in S_{HH}(r_2),$ and $(a_2, r_2) \in admin-authority$
$a_1$ assign $u_2$ to $r_1$	
Supported, because: $can\_assign(a_1, r_2, r_1)$	Supported, because: $can\_assign(a_1, r_2, r_1)$
$a_2$ assign $p_2$ to $r_4$	
Supported, because: $can\_assignp(a_1, r_2, r_4)$	Supported, because: $can\_assignp(a_1, r_2, r_4)$

In the mean time, several researchers have found the limitations of the monotype hierarchy. Li *et al.* [9] have analyzed the standard of RBAC model and suggested that the monotype hierarchy in the standard has ambiguities. Sandhu *et al.* [13] have emphasized the necessity of using two different hierarchy relations by comparing the RBAC model with the LBAC (Lattice-based access control) model. Moffett *et al.* [10] further show that there are three different uses (semantics) of a role hierarchy, which imply that a monotype hierarchy is not sufficient. Finally Joshi *et al.* [8] have proposed a formal definition of hybrid hierarchy, which consists of *I*-hierarchy, *A*-hierarchy and *IA*-hierarchy. The hybrid hierarchy can support three different semantics and can support much more fine-grained constraints.

To the best of our knowledge, little research has addressed the issue of how to build a complete administration model for an RBAC system with hybrid hierarchy, which is the primary goal of our paper.

## 7 Conclusion and Future Work

In this paper, we extend our previous work (SARBAC07) and propose the SARBAC-HH model that can not only deal with the hybrid hierarchy but also support much more fine-grained administration semantics. Our model borrows the key concept of administrative scope from Crampton *et al.*'s SARBAC model and redefines it in presence of hybrid hierarchy. In SARBAC-HH, we refine SARBAC07's Role Hierarchy Administration model (RHA07) and define more fine grained semantics for the administration operations. We propose a family of RHA-HH models incrementally for the RBAC-HH model. The RHA-HH<sub>1</sub> model contains the basic elements for administering a hybrid hierarchy. The RHA-HH<sub>2</sub> model assigns administration permissions to each role. The RHA-HH<sub>3</sub> model introduces the special administrative roles. The RHA-HH<sub>4</sub> model adds the update features for the administrative roles. Finally, we evaluate our model in detail and compare our proposed SARBAC-HH model with the ARBAC07 model. We show that the SARBAC-HH model has

the advantages such as flexibility and simplicity over ARBAC07

In the future, we plan to add the time constraint into consideration and build a complete administration model for GTRBAC using hybrid hierarchy. We also plan to implement the idea in this paper and get some empirical results.

## Acknowledgment

Please provide acknowledgement only after the conclusion section.

## 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. Ferraiolo, 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. , 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 1<sup>st</sup> 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, 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* (in press).
- [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): 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 15<sup>th</sup> Computer Security Applications Conference, Arizona, US, Feb 1999, pp. 229-238.
- [16] B. Shafiq, J. B. D. Joshi, E. Bertino, A. Ghafoor, "Secure Interoperation in a Multi-Domain Environment Employing RBAC Policies", *IEEE Transactions on Knowledge & Data Eng.*, vol. 17, no. 11, pp. 1557-1577, Nov. 2005.
- [17] Y. Zhang, and J. B.D. Joshi, "SARBAC07: A Scoped Administration Model for RBAC with Hybrid Hierarchy", In Proceedings of 3<sup>rd</sup> International Symposium on Information Assurance and Security, pp.149-154, Aug. 2007
- [18] Yue Zhang and James, B.D. Joshi, "ARBAC07: A Role-Based Administration Model for RBAC with Hybrid Hierarchy", In Proceedings of Information Reuse and Integration (IRI' 2007), Las Vegas, NV